WEST

Generate Collection

L2: Entry 1 of 2 File: USPT May 25, 1999

US-PAT-NO: 5907704
DOCUMENT-IDENTIFIER: US 5907704 A

TITLE: Hierarchical encapsulation of instantiated objects in a multimedia authoring system including internet accessible objects

DATE-ISSUED: May 25, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Gudmundson; Norman K.	Half Moon Bay	CA	N/A	N/A
MacInnis; Bo Yu	San Carlos	CA	N/A	N/A

ASSIGNEE INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Quark, Inc.	Denver	CO	N/A	N/A	02

APPL-NO: 8/ 720660
DATE FILED: October 2, 1996

PARENT-CASE:
CROSS REFERENCE TO RELATED APPLICATIONS This is a continuation-in-part of the application, "Hierarchical Encapsulation of Instantiated Objects in a Multimedia Authoring System," Ser. No. 08/415,848, filed Apr. 3, 1995, also assigned to mFactory, Inc., now U.S. Pat. No. 5,680,619.

INT-CL: [6] G06F 9/40

US-CL-ISSUED: 395/701
US-CL-CURRENT: 717/1

FIELD-OF-SEARCH: 395/701

REF-CITED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5287447	February 1994	Miller et al.	345/342
☐	5423041	June 1995	Burke et al.	395/705
☐	5487141	January 1996	Cain et al.	345/435
☐	5493680	February 1996	Danforth	395/702
☐	5499333	March 1996	Doudnikoff et al.	707/515
☐	5680619	October 1997	Gudmundson et al.	395/701

ART-UNIT: 274

PRIMARY-EXAMINER: Voeltz; Emanuel Todd

ASSISTANT-EXAMINER: Chavis; John Q.

ATTY-AGENT-FIRM: Webb; Glenn L.

ABSTRACT:

An application development system, optimized for authoring multimedia titles, enables its users to create selectively reusable object containers merely by defining links among instantiated objects. Employing a technique known as Hierarchical Encapsulation, the system automatically isolates the external dependencies of the object containers created by its users, thereby facilitating reusability of object containers and the objects they contain in other container environments. Authors create two basic types of objects: Elements, which are the key actors within an application, and Modifiers, which modify an Element's characteristics. The object containers (Elements and Behaviors--i.e., Modifier containers) created by authors spawn hierarchies of objects, including the Structural Hierarchy of Elements within Elements, and the Behavioral Hierarchy, within an Element, of Behaviors (and other Modifiers) within Behaviors. Through the technique known as Hierarchical Message Broadcasting, objects automatically receive messages sent to their object container. Hierarchical Message Broadcasting may be used advantageously for sending messages between object containers that may be located remotely from each other, such as over a Local Area Network or the Internet. Even whole object containers may be transmitted and remotely recreated over the network. Furthermore, the system may be embedded within a page of the World-Wide Web.

5 Claims, 87 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 55

BRIEF SUMMARY:

I. BACKGROUND

A. Field of the Invention

This invention relates to application development systems generally, and in particular to systems for authoring interactive multimedia applications, including applications able to communicate over networks, and applications embedded into pages on the World Wide Web.

B. Description of the Related Art

Since the advent of computers many decades ago, computer scientists have labored to build increasingly powerful computer hardware that is faster and cheaper to build than its predecessors. Today's computers can perform, in one second, many millions of "add," "shift," "load," "store" and other relatively simple functions.

To perform tasks of any significant complexity, however, one must somehow cause computers to perform vast numbers of these simple functions in some sequence. Computer software, in its most basic form, comprises programs or sequences of instructions, each of which directs a computer to perform the function corresponding to that instruction.

Yet, even as computers have become more powerful, developers of computer software continue to struggle to create complex programs without having to "reinvent the wheel" for each task they direct computers to perform. This need for "reusability" permeates virtually every aspect of software development, and is driven ultimately by the end user's desire for "ease of use."

1. Reusability and the Modular Interface

Although there are many varied approaches to this basic problem of reusability, one technique remains constant--"modular programming," i.e., the "bootstrapping" of multiple simple functions into "modules" of greater and greater complexity that can be reused through higher-level "modular interfaces." Virtually all software relies on modular programming to some extent.

For example, assemblers and compilers enable programmers to bootstrap from the machine language defined by a computer's instruction set to a higher-level, more human-readable language. Similarly, operating systems perform many of the low-level support tasks commonly needed by applications software developers (e.g., file system, memory management, basic GUI routines, etc.), and then provide developers with a library of "reusable" modules.

In essence, all application development systems provide to their users a language, an application programming interface ("API"), or some other form of modular interface designed to facilitate development of general purpose or highly specialized applications. Such modular interfaces provide reusability of the "hidden" functionality which implements that interface.

2. Encapsulation and the Object Interface

One of the most popular current trends designed to promote the creation of reusable software is in the field of object-oriented programming ("OOP"). There are a variety of OOP languages and application development tools on the market today (e.g., C++ and Kaleida Labs' "ScriptX.TM."), as well as a number of "pseudo-OOP" tools (e.g., "HyperCard.TM." from Apple Computer and "Visual Basic.TM." from Microsoft) which borrow some, but not all, of the basic principles of OOP.

OOP systems generally are intended to extend reusability to all portions of a computer program, not just to particular modules. This is accomplished by distributing control of the various tasks performed by a program to individual "objects" with well-defined modular interfaces. These objects typically communicate by directly calling one another's capabilities or "methods."

OOP application development systems generally require their users to define abstract "classes" of objects that define the characteristics of the objects in that class. The actual objects in the user's application are instantiated as needed at "runtime" from this class "template." Such systems also typically include certain built-in reusable class libraries to provide the user with an initial base of functionality.

The instantiated objects that perform the various tasks within the user's application are reusable in large part due to a process known as "encapsulation." Encapsulation involves defining a class of objects with a well-defined external interface, and "hiding" within the objects the code and data necessary to implement that interface. In other words, the implementation of an object's interface is "encapsulated" entirely within the object.

Thus, programmers can reuse objects in various different contexts within their application. All other objects that are aware of this interface can "reuse" this object because its external dependencies are isolated within its modular "object interface."

Programmers also can use a process known as "inheritance" to "specialize" or modularly extend an object's functionality by reusing some or all of that object's characteristics via its interface or class definition--i.e., the public methods and data structures that define the template for instances of that class. For example, programmers may wish to extend the base functionality of an existing class (whether created by the programmer or provided in a reusable class library) by adding a new method to objects in that class, or replacing (i.e., overriding) or supplementing (i.e., overloading) an existing method.

By defining a sub-class of objects that "inherit" the characteristics of the "parent" class, the programmer can reuse (as well as add to, replace or supplement) those inherited characteristics. For example, a "car" or "bus" sub-class might inherit characteristics from a more general "vehicle" class. A car is a "kind of" vehicle, and may inherit, for example, a vehicle's color and speed properties, as well as steering and braking methods.

3. The Need for Selective Reusability of "User Objects"

Despite their facilities for reusability of objects, OOP application development systems remain difficult to use. It is far simpler for application developers to create and configure instantiated objects directly, as opposed to defining abstract classes each time they need to create a slightly different object or to model the relationships or interaction among existing objects.

This problem is exacerbated when an application developer creates and desires to reuse compound or aggregate "user objects"--i.e., a group of objects that are related in the context of a particular application, but do not necessarily share common characteristics. These "user objects," in addition to requiring extensive inter-object communication, often involve "part of" or "container" relationships that occur frequently in any modularly designed application.

Inheritance and encapsulation are well-suited to model "kind of" relationships to further specialize the atomic objects from which an application will be built. All modular systems require certain atomic "building blocks" which are reusable and provide sufficient performance, even if difficult to modify.

To create applications of any significant complexity, however, one must combine these atomic objects into more complex "user objects" that interact with one another. It is this common modular process that illustrates the significant limitations on reusability imposed by traditional OOP systems.

OOP systems allow developers to "reuse" an existing class of objects by: (i) creating a class of objects that inherits from and "specializes" that existing class; (ii) creating a class of objects that encapsulates that existing class within its private (hidden) data structures and methods; or (iii) creating a distinct class of objects that communicates with (i.e., invokes methods of) that existing class of objects.

Yet, in all three cases, the new class of objects is tightly coupled to, and thus highly dependent upon, the existing class of objects. Mere communication among otherwise unrelated objects provides virtually no reusability, and creates many explicit dependencies. Although inheritance and encapsulation provide reusability of objects at various levels of complexity, a class of objects at any given level of complexity remains highly dependent upon the particular characteristics it encapsulates or inherits from its less complex superclasses.

It therefore remains quite difficult, if not impossible, to "selectively reuse" characteristics from (and interaction among) complex "user objects" in new environments. One cannot, for example, easily "mix and match" object characteristics across class libraries and choose "some from column A and some from column B." At any given level of complexity, a developer cannot simply replace undesired characteristics of a complex object from one class library with the more desirable characteristics of another complex object from a different class library. Until such selective reusability is achieved, the ultimate promise of OOP cannot be considered fulfilled.

a. An OOP Windowing Example.

Consider the following example involving simple windowing functionality, a very common application for traditional OOP systems. Imagine two independent developers ("A" and "B") creating windowing systems in a typical OOP development environment.

Developer A might create a "Window" class of window objects with a name ("Window X"), a particular size (specified by a bounding rectangle), a title bar (specified by another bounding rectangle), a color and a beveled border. Its only functionality might be to create, destroy and draw these window objects, via three respective methods.

To enhance the functionality of these window objects in a modular manner, Developer A might create a "Minimize" subclass that inherits from the Window class and adds a "minimize" capability. When a user clicks on the small minimize box in the upper right hand corner of the window, the minimize object transforms the window object into a small icon at the bottom of the screen,. When the user

clicks on the icon, the window is restored.

This Minimize subclass would require "minimize" and "maximize" methods to perform these functions, as well as a minimize box (specified by a bounding rectange), a minimize flag (indicating the state of the window--i.e., whether it is currently minimized), and data for a minimize icon (pattern, size, position, etc.). In addition, the Minimize subclass would overload the create, destroy and draw methods of the Window class--i.e., to create, destroy and draw the minimize box and/or icon as well as the window.

Finally, Developer A might add additional "drag" functionality to these window objects, which would enable the user to drag the window by its title bar. This subclass could inherit all data and methods from the Minimize subclass (which itself inherits from the Window class), and simply add a "drag" method.

Developer B independently might take a similar approach. But, the class of Window objects created by Developer B (named Window X') might not include the beveled border. Moreover, Developer B might add a "window shade" capability that differs slightly from the minimize capability created by Developer A. When the user "double clicks" on the title bar of Window X', the Window might "roll up," showing only the title bar. Developer B might implement this capability by creating a "Window Shade" subclass that inherits from (and overloads the methods of) the Window class, and adds a drag method to implement this "window shade" capability. That subclass might also include a flag to track whether the window currently is "pulled up."

Assume that Developers A and B distribute their respective object classes in class libraries containing the public interface (i.e., the methods and data identified above), but not the private code and data that implements that interface. Selective reusability would allow Developer "C" to reuse all or any modular portion of the functionality created by Developers A and B, without recompiling these classes or obtaining access to any such private code.

It is true that Developer C could reuse the Window class, the Minimize subclass or the Drag subclass, thereby reusing functionality at various levels of abstraction. But, what if Developer C wanted to reuse the window and the drag capability created by Developer A, but integrate the window shade capability of Window X' in lieu of the minimize capability of Window X?

In short, Developer C cannot easily accomplish this task. The problem is that the private data and methods of the Minimize and Drag subclasses are dependent upon the public data and methods of the classes from which they inherit. The code might not even link.

The Drag subclass inherits from, and thus assumes the existence of, the Minimize class. The drag method might, for example, modify the bounding rectangle of a Minimize object to "move" that object (as well as the rest of the window) across the screen as the user drags the window. This dependency effectively prevents Developer C from replacing the Minimize class with the Window Shade class. The drag method would attempt to modify data that no longer exists.

Developer C would be forced to rewrite the Window Shade and Drag classes (potentially a significant amount of code) merely to reuse the Window class. Reusability therefore is substantially impaired by the external (inter-class or inter-object) dependencies of a subclass on its superclass. As noted above, encapsulating these dependencies or invoking other objects' methods within a class' private methods (as opposed to relying on inheritance) creates similar dependencies and perhaps even less reusability.

Any system of reasonable complexity will contain many levels of these external dependencies, making selective reusability of complex "user objects" across class libraries extremely difficult, if not practically impossible. As discussed below, today's application development systems create similar external dependencies that impair selective reusability.

4. The Lack of Selective Reusability in Multimedia Authoring Systems

Consider, for example, the field of multimedia authoring systems to which an embodiment of the present invention is directed. Although there exist many such

systems with almost as many different techniques, these systems all impose significant limitations on the author's ability to selectively reuse complex "user objects" across different environments.

Certain systems, such as Macromedia's Director,.TM. employ traditional metaphors with which authors are familiar and quite comfortable. Director employs a frame-based metaphor that requires authors to determine precisely which objects will be present in each frame of a sequence. This metaphor is familiar to animators, and frequently is used for constructing sequential animation sequences containing relatively little interactivity.

Constructing highly interactive applications, however, is quite difficult within the confines of a frame-based environment. Even with a scripting language included to provide greater control over the sequence in which characters appear, interactivity frequently is limited to that provided by one or more scripts within a frame.

It is quite difficult to reuse any of the characteristics of a particular object, because the object is highly dependent upon its environment (i.e., the scripts of other objects that "call" it, as well as the confines of the frames in which it appears, relative to other objects). Extensive reliance on scripting to model interactivity makes it difficult to isolate an object's external dependencies within a single script, and virtually impossible to isolate such dependencies across multiple scripts.

Rigorous OOP tools (such as the ScriptX.TM. language from Kaleida Labs and the more visually oriented Quest.TM. from Allen Communication) still require significant programming expertise on the part of their users. Despite offering extensive libraries of reusable "high-level" objects, such products remain relatively low-level development tools--i.e., OOP programming languages (or perhaps visual programming languages), as opposed to authoring tools.

As noted above, such tools impair selective reusability by creating external (inter-object) dependencies as a result of inheritance and encapsulation. Visual programming techniques facilitate the author's task of creating complex applications, but still frequently require scripting and/or programming to modify an object's characteristics or to model inter-object communication. The visual interface, at best, masks the programming that controls the user's application beneath the surface. In any event, the external dependencies remain, and reusability is impaired.

Other systems employ "pseudo-OOP" techniques in an attempt to combine the power of reusable objects with the flexibility of customizing the characteristics of particular objects and modeling inter-object communication. Forms-based authoring systems, such as Apple Computer's HyperCard.TM. and Microsoft's Visual Basic.TM., provide authors with significant freedom to create and configure individual objects using highly visual interfaces. Yet, such systems still rely heavily on scripting to implement an object's functionality, as well as inter-object communication and other forms of interaction.

Though these systems often contain many built-in types of objects, it is extremely difficult to create new object types. An object's unique characteristics therefore are determined by its script. Because objects communicate with one another directly via scripts, these scripts are highly dependent upon one another, and cannot easily be reused in other environments. Moving an object from one environment to another generally requires reading the scripts not only for that object, but for other objects in its former environment.

One category of products (e.g., Apple Computer's Apple Media Tool.TM.) provides a highly visual approach to authoring with virtually no scripting or programming involved. In other words, as opposed to visual programming languages (which, in essence, are merely "easy to use" scripting languages), such products provide true "object-based authoring"--i.e., the author creates and configures actual instantiated objects (or pseudo-objects) with little or no scripting or programming.

One problem with such tools is that they are extremely limited and inflexible. Authors typically cannot create custom events or messages, much less add

functionality to an existing object or group objects together in any meaningful way. Moreover, such systems provide no mechanism for modeling modular "container" relationships among objects, and reusing these more complex "user objects" in different container environments.

All of the approaches noted above suffer from a lack of support for the creation of complex "user objects" that can be selectively reused in other environments. An object is only reusable to the extent that its dependencies on its external environment are isolated within (i.e., known to) that object. An object is only selectively reusable to the extent that it is loosely coupled to the objects it contains, thereby permitting authors to modify this relationship.

What is needed is a system that models this "container" relationship among objects in a manner that permits authors to selectively reuse object containers and the objects they contain across different container environments.

5. The Lack of an Object Communications Hierarchy in Multimedia Authoring Systems

A further problem arises when authors elect to design multi-user applications or applications that interact with one another. In such cases, there is a need for components of one application to interact or communicate with components of another application. The difficulty lies not only in providing a general mechanism which facilitates interapplication communication, but in minimizing or eliminating the external dependencies of the communicating components in each application.

Various mechanisms currently exist for inter-application communication ("IAC") among applications located on a single computer or distributed across a local area network ("LAN") or even the Internet. For example, Apple Events.TM. for the Macintosh.TM. operating system and OLE.TM. for the Microsoft Windows.TM. operating system both provide a general purpose mechanism that permits "objects" in one application to communicate with, or at least trigger functionality provided by, "objects" in another application.

Moreover, client-server applications typically provide a communications interface that enables a "client" application to issue a request to a "server" application to initiate a process on the server and perhaps return information as a result. The recent explosion in popularity of the Internet, and in particular the World Wide Web, has resulted in a surge of client-server applications.

Many are based on traditional networking protocols, such as the HyperText Transfer Protocol ("HTTP") and the File Transfer Protocol ("FTP"), which enable client applications to retrieve an item located on a remote server by specifying a unique identifier for that item, known as a Uniform Resource Locator ("URL"). Thus, a client World Wide Web browser (such as the Netscape Navigator.TM. or Microsoft Internet Explorer.TM.) can retrieve an HTML page from an HTTP server, or a program or other file from an FTP server, by issuing the appropriate URL containing the HTTP or FTP request (e.g., "http://www.company.com/page.html" or "ftp://ftp.company.com/file.exe").

Other client-server and distributed ("peer-to-peer") applications include multi-player games containing customized inter-application communications interfaces built with traditional programming languages such as C and C++. These games typically include identical client applications that communicate with one another directly, or through a server acting as a communications intermediary. Newer object-oriented programming environments, such as Sun Microsystems Java.TM. programming language and the Microsoft ActiveX.TM. component API, enable programmers to develop and deliver over the Internet interactive client applications which can communicate with other such client applications in a similar manner.

One problem inherent in all of the above-mentioned IAC mechanisms is the lack of support for communication among complex "user objects" as discussed in the previous section. Using such mechanisms, applications can communicate with one another only at the "object-to-object" level. As programmers develop more complex functionality by combining objects together that interact with one another, there is a need for these "user objects" to communicate (both within

and across applications). As noted above, the lack of a mechanism to loosely couple objects into more complex "user objects" results in the creation of external dependencies and a lack of selective reusability of these "user objects." These same dependencies that exist within an application also exist across applications that utilize mere "object-to-object" communication schemes provided by existing IAC mechanisms.

Moreover, existing IAC mechanisms provide only for "point-to-point" messaging--i.e., messages sent from one object in one application to another object in another application. A programmer desiring to send a message to another application must know precisely which object in that other application will respond to the message. Such stringent requirements severely limit the flexibility of communication among applications, particularly if they are created by different programmers. What is needed is a hierarchical messaging infrastructure which enables programmers to target their messages at any known point in the other application's hierarchy, and which facilitates the handling of that message by objects below the targeted object in the hierarchy.

II. SUMMARY OF THE INVENTION

The present invention encompasses an application development system that enables its users to create reusable "object containers" merely by defining links among instantiated objects. Employing a technique referred to herein as Hierarchical Encapsulation, the system automatically isolates the external dependencies of the object containers created by its users. This isolation of external dependencies resolves the problems addressed above regarding selective reusability of "user objects," thereby facilitating the development of applications of increasing complexity.

Objects contained within other objects are not "hidden" within or tightly coupled to their object container environments. Rather, they are loosely coupled with those environments, and therefore can more easily be reused in other environments. By virtue of being contained within another object, the contained object automatically is afforded access to its environment. Its object container is, in essence, an "environmental frame of reference" for the objects it contains. For example, unless overridden by the author, objects automatically receive messages sent to their object container. They automatically can access data known to their object container. Their position is even determined relative to their object container.

Moreover, objects are decoupled from their characteristics. By defining two distinct types of objects (one of which modifies the characteristics of the other), and loosely coupling (i.e., temporarily linking) these two types of objects, the system provides a mechanism for authors to modify an object's characteristics merely by deeming one object to be contained within another. Removing that object from its container removes that characteristic. In this manner, authors easily can modify an object's characteristics and reuse it in other environments.

In one embodiment described herein, the system is optimized for the development of interactive multimedia applications or "titles." This multimedia authoring system provides its users ("authors") with a visual authoring interface that requires little, if any, scripting or programming. The system employs a form of object-based authoring in which authors create and configure instantiated objects directly, typically by "dragging and dropping" icons and configuring dialog boxes.

Authors can create two basic types of objects: Elements and Modifiers. Elements represent the actual characters or actors that interact with one another in the author's title. Elements generally can be linked to external media (such as text, sounds, pictures, animations and movies), and possess certain inherent characteristics relating to that media.

Authors can supplement an Element's inherent characteristics by incorporating Modifiers within that Element. These Modifiers provide the Element with properties (known as Variables) that further define what the Element is and capabilities that further determine what the Element does. A special type of Modifier, known as a Behavior, can contain additional Behaviors and other Modifiers, providing the author with a mechanism to create a complex Element

"personality."

Both Elements and Behaviors are "object containers"--in this embodiment, object instances that can "contain" (i.e., be linked to) other object instances. Elements can contain Modifiers as well as other Elements; and Behaviors can contain Modifiers, including other Behaviors.

By incorporating Elements within Elements, authors create a Structural Hierarchy of Elements, each Element providing an environmental "frame of reference" for the Elements it contains. These "parent" Elements enable authors to provide structure for their titles and to model relationships among their Elements.

Elements can communicate with one another at a high "Element level," without regard to their child Elements. In one respect, Elements "encapsulate" their child Elements by creating a modular interface through which an Element's child Elements can communicate with objects external to that Element container.

Similarly, by incorporating Behaviors (and other Modifiers) within Behaviors, all inside an Element, authors create a Behavioral Hierarchy within the Element--i.e., the Element's internal "personality." Within the context of an Element "personality," each Behavior provides an environmental "frame of reference" for the Modifiers it contains. These "parent" Behaviors enable authors to model the relationships among the various Behaviors within an Element's overall personality.

Elements, in effect, "inherit" the characteristics provided by their internal Behavioral Hierarchy. Because Elements and Modifiers are distinct, loosely coupled objects, authors can modify an Element's characteristics merely by adding Modifiers to (or removing Modifiers from) an Element.

The system provides for significant reusability of object containers by utilizing the Structural and Behavioral Hierarchies to isolate the external dependencies of Elements and Behaviors. In essence, the system automatically "encapsulates" an author's object containers. Once encapsulated, they can be reused in other "environments." Moreover, by loosely coupling an Element to the Modifiers it contains, the system enables authors to modify their Elements so as to "inherit" and "disinherit" characteristics while maintaining an evolving hierarchical encapsulation vis-a-vis the Element's external environment.

Using a technique known as Adoption, an author can cause an Element to be "adopted" by a new parent Element. Using a similar technique known as Transplantation, an author can "transplant" an Element's Behavior (or its entire "personality") into another Element.

Because Hierarchical Encapsulation is integrated into the Structural and Behavioral Hierarchies determined by the author's object containers, authors obtain the benefits of this technique automatically. Their Elements and Behaviors are thus selectively reusable.

For example, a mechanism known as Hierarchical Message Broadcasting provides a structured messaging system that broadcasts messages from their initial destination down the Structural and Behavioral Hierarchies to all descendant Elements and Modifiers. This mechanism isolates an object container as a centralized abstract destination for all messages intended for "any object within that object container." This mechanism facilitates reusability of object containers in other environments in that an object container's new "parent" Element will provide it with messages automatically.

Another mechanism, known as Hierarchical Variable Scoping, makes a Variable accessible automatically to all descendant objects of the Variable's parent Element or Behavior. This mechanism isolates an object container's dependencies on Variables that are external to that object container, but still within its ancestral "environment." By making such Variables "known" to those objects in the object container that rely on that Variable, the object container can be moved to another environment with a well-defined external interface that "knows" which external Variables are assumed to be present in that environment.

Yet another mechanism, known as Hierarchical Relative Positioning, determines the position of a child Element relative to the position of its parent Element.

As a result, the child Element moves with its parent Element automatically. This mechanism isolates an Element's external positional dependencies--i.e., the effects of an Element's environment on the Element's position.

In addition to the "built-in" Elements and Modifiers, the system is quite extensible via a "Component API." This Component API enables programmers to seamlessly integrate new Modifiers (and "Services" that support them) into the system.

The system also provides an inter-application communication mechanism integrated into the system's Hierarchical Message Broadcasting mechanism. By providing a Net Messenger Modifier and corresponding Service, an author can target a message to any known Element in another Project/title (as well as within its own Project/title), and utilize the Structural and Behavioral Hierarchies to propagate that message to descendant Elements and Modifiers. As is the case with intra-Project/title messaging, this mechanism facilitates reusability of object containers in other environments by providing them with messages via their new "parent" Elements.

In addition to facilitating selective reusability of object containers, this communication mechanism also provides an infrastructure for targeting and propagating messages in accordance with the object hierarchies of each individual Project/title.

Rather than limiting communication to "point-to-point" messaging between objects, the system utilizes for inter-application communication the very same object hierarchies that each author constructs for intra-application communication. In this manner, one can target a message to an object container in another application without knowing which object inside that object container will handle the message. The target application's Structural and Behavioral Hierarchies will determine how the message is handled. The authors only need agree on the name and meaning of messages, without having to worry about the name or even functionality of the objects that ultimately will handle such messages.

The system's inter-application messaging mechanism enables authors to send to another Project/title not only messages, but also data attached to those messages. Such data can range from simple integers, strings and other common data types to more complex lists, compound objects and even entire object containers including their Structural and Behavioral Hierarchies. In the latter case, these "teleported" object containers are much more than mere "data" and must be instantiated in the target Project/title. Just as Elements within a Project/title can dynamically be "cloned" or "killed", and "adopted" by (or have their Modifiers "transplanted" to) another "parent" object container, so too these "teleported" object containers must be "cloned" or instantiated in the target Project/title and "adopted" by or "transplanted" to their new target "parent" object container (e.g., the initial destination of the inter-application message).

By utilizing "Object Reference Variables," an author can specify the initial destination of a message to be an object's relative or absolute point within the Structural and/or Behavioral Hierarchies (within and across Projects/titles). Authors can leverage their degree of knowledge of the Structural and Behavioral Hierarchies in the target Project/title by targeting a message to any known object container (including the highest "Project" level). By resolving such references dynamically, the system provides authors significant flexibility in targeting objects that might move within their own Structural and Behavioral Hierarchies.

"List Variables" provide authors with a convenient mechanism for, among other things, targeting multiple remote Projects/titles. This mechanism is extremely useful in the context of multi-user titles in which the number of users can change dynamically.

The ability to specify a list or array of remote hosts enables the author to target a message to all hosts with a single Net Messenger Modifier (as is usually desired), without sacrificing the ability to target one or more individual hosts when necessary.

Also, an author can embed a title within a World Wide Web page by utilizing a browser plug-in and the "Open URL Modifier." With this mechanism, the author can provide a user (within the context of a browser window) the ability to interact with multiple titles. Any Element can link dynamically to another title (or open any page on the World Wide Web) as easily as it can respond to any runtime message sent within or across titles.

Finally, the architecture of the system is substantially platform-independent. Titles can be "played" on multiple platforms. Moreover, the entire authoring environment can be ported to a variety of platforms with relatively little modification due to the isolation of a platform-dependent layer within the system.

DRAWING DESCRIPTION:

III. BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is an illustration of the dual hierarchy (Structural and Behavioral) underlying the principles of the present invention.

FIG. 2 is a screen display showing the layout view window, tool palette, object information palette and menus of the graphical user interface of the implementation of one embodiment of the present invention.

FIG. 3 is screen display showing the layers view window of the graphical user interface under one embodiment of the invention.

FIG. 4 is a screen display showing the structure view window of the graphical user interface under one embodiment of the invention.

FIG. 5 is a screen display showing the asset palette of the graphical user interface under one embodiment of the invention.

FIG. 6 is a screen display of showing a library window in the graphical user interface under one embodiment of the invention.

FIG. 7 is a screen display showing the alias palette of the graphical user interface under one embodiment of the invention.

FIG. 8 is a screen display showing the messaging log window of the graphical user interface under one embodiment of the invention.

FIGS. 9(a)-(d) are screen displays showing the mToon editor windows of the graphical user interface under one embodiment of the invention.

FIG. 10 is a screen display showing the Element configuration dialog of the graphical user interface under one embodiment of the invention.

FIG. 11 is a screen display of a Behavior configuration dialog in the graphical user interface under one embodiment of the invention.

FIGS. 12(a)-(j) are screen displays showing the Modifier configuration dialogs for Variables in the graphical user interface under one embodiment of the invention.

FIGS. 13(a)-(g) are screen displays showing the Modifier configuration dialogs for Messengers, including a Net Messenger, as well as a Net Messaging Service and an Open URL Modifier in the graphical user interface under one embodiment of the invention.

FIGS. 14(a)-(c) are screen displays showing the Modifier configuration dialogs for Scene-based Modifiers in the graphical user interface under one embodiment of the invention.

FIG. 15 is a diagram showing the operation of the Scene change Modifier of FIG. 14(a) in conjunction with the return Scene Modifier of FIG. 14(b).

FIGS. 16(a)-(d) are screen displays showing the Modifier configuration dialogs

for Motion Modifiers in the graphical user interface under one embodiment of the invention.

FIGS. 17(a)-(d) are screen displays showing the Modifier configuration dialogs for Graphics Modifiers in the graphical user interface under one embodiment of the invention.

FIGS. 18(a)-(b) are screen displays showing the Modifier configuration dialogs for Sound Modifiers in the graphical user interface under one embodiment of the invention.

FIG. 19 is a screen display showing the Modifier configuration dialog for the style Modifier in the graphical user interface under one embodiment of the invention.

FIG. 20 is a screen display showing the Modifier configuration dialog for the cursor Modifier in the graphical user interface under one embodiment of the invention.

FIG. 21 is a screen display showing the Modifier configuration dialog for the set Miniscript Modifier in the graphical user interface under one embodiment of the invention.

FIG. 22 is a screen display showing the Modifier configuration dialog for the classification Modifier in the graphical user interface under one embodiment of the invention.

FIG. 23 is a screen display showing the Modifier configuration dialog for the set value Modifier in the graphical user interface under one embodiment of the invention.

FIG. 24 is a high level block diagram of the implementation of one embodiment of the present invention.

FIG. 25 is a diagram showing the interconnection of the various classes in the object model of the present invention.

FIG. 26 is a diagram showing the interconnection of the Service classes in the object model of the present invention.

FIG. 27 is a diagram showing the storage layout for a multimedia title in one embodiment of the present invention.

FIG. 28 is a flowchart showing the event processing loop under one embodiment of the invention.

FIG. 29 is a diagram showing the architecture of both the Component API and authoring GUI according to the present invention.

FIG. 30 is a diagram showing the operation of the object model of the present invention.

FIG. 31 is a flowchart showing the loading of Components (Modifiers, Services) at boot-up time of the program under one embodiment of the invention.

FIG. 32 is a screen display showing an implementation of a snake.

FIG. 33 is a screen display showing an implementation of a school of fish.

FIGS. 34(a)-(g) are screen displays of a selectively reusable windowing system.

FIGS. 35(a)-(g) are diagrams showing the implementation of the selectively reusable windowing system.

FIGS. 36(a)-(b) are, respectively, a screen display showing an implementation of an Internet Tic Tac Toe game, and a high-level block diagram showing the key inter-Element and internetwork messages for the game.

FIGS. 37(a)-(b) are screen displays of the Structural Hierarchy of the Internet

Tic Tac Toe game.

FIGS. 38(a)-(g) are screen displays of the various Behaviors used to implement the Internet Tic Tac Toe Game.

FIG. 39 is a diagram illustrating the operation of an Internet "Ultra Chess" game.

DETAILED DESCRIPTION:

IV. DETAILED DESCRIPTION

A. External Architecture--"Author's-Eye View"

Before discussing the underlying implementation of the authoring system of the present invention, it is helpful to examine its external architecture--i.e., that portion of the system which is visible to the author.

It is through this external architecture that authors create end-user applications or "titles." As is explained in greater detail below, programmers can extend the functionality of this external architecture, seamlessly, via an interface to the "core" of the system. Moreover, the platform-independent nature of the system's implementation facilitates portability of the efforts of authors and programmers to a wide variety of computing platforms.

1. Objects: Elements and Modifiers

In one embodiment of this invention, the author can create two basic types of objects: Elements and Modifiers. To use the metaphor of a stage play, the Elements comprise the play's structure (e.g., acts and scenes) and its actors, and the Modifiers comprise the stage directions, the motivations the director imbues in the actors to behave the way they do, and other modifications to the characteristics of the play's structure and its actors.

The system enables authors to create various types of Elements and to assign particular characteristics to each Element. These characteristics include properties (e.g., weight, color, etc.) that define what the Element is and capabilities (e.g., send messages, move along a path, etc.) that determine what the Element does.

Authors assign characteristics to an Element by incorporating Modifiers within that Element. As is illustrated below, the process of creating and configuring Elements and Modifiers is highly visual and intuitive, typically involving "dragging and dropping" icons representing Elements and Modifiers, selecting menu items and configuring dialog boxes. Little, if any, scripting or programming is required.

The system is, however, quite extensible via the "Component API," discussed in greater detail below. Programmers can use the Component API to create "Components" that become fully integrated Modifiers and "Services" (which "service" Modifiers but are not accessible directly to authors). Once created, Components are indistinguishable from the system's "built-in" Modifiers and Services.

In this embodiment, there are three general categories of Modifiers: (i) Variables that simply store data of various types; (ii) Capabilities that perform actions on behalf of the Element; and (iii) Behaviors that "contain" additional Behaviors and other Modifiers. In another embodiment, a hybrid Variable/Capability Modifier could both store data and perform actions.

An author can create two general categories of Elements: (i) Structural Elements (including Projects, Sections and Subsections, discussed in greater detail below) that serve only to "contain" other Elements and Modifiers; and (ii) Media Elements (including Scenes, discussed below) that not only can contain other Media Elements and Modifiers, but also can be "linked" to raw media, such as text, sounds, pictures, animations and movies.

There are three basic types of Media Elements that an author can create: (i)

Graphic Elements (including still pictures, animations and videos), Text Elements and Sound Elements. An author can then link raw media of a particular format to the Element, causing that Element to attain certain inherent capabilities (e.g., to make a sound, display a picture, or play a movie).

By linking raw media of a particular format to a Graphic, Text or Sound Element, the author causes the system to "morph" the Element into a more specific type of Element (and among specific types) related to that format. For example, the author could create an "AIFF" sound, an "ASCII" sentence, a "PICT" picture, a "QuickTime" movie or an "mToon" (a built-in animation format). The morphing process is described in greater detail below with respect to the system's implementation.

Elements of a given type also have certain built-in properties known as Attributes. An Element's Attributes include properties applicable to the type of media to which the Element is linked (e.g., position, cel number, rate, etc.).

Thus, an Element's properties are defmed by its Attributes and its author-created Variables. An Element's capabilities are determined by the type of media, if any, to which it is linked, and by its author-created Behaviors and Capabilities (i.e., its non-Variable Modifiers). Together, these properties and capabilities define an Element's individual characteristics--i.e., what the Element is and what the Element does.

In addition to creating Elements and assigning them individual characteristics, an author can specify the manner in which these Elements interact with one another, via an integrated object messaging mechanism. This messaging mechanism is accessible to authors via Modifiers, as is described in greater detail below.

2. Object Containers: Elements and Behaviors

Unlike simple "leaf" objects, Elements and Behaviors are "object containers" that can comprise a hierarchy of objects. These object hierarchies provide "environments" (each enclosed by an object container) in which the system can isolate external dependencies. This isolation of an object container's external dependencies, referred to herein as "Hierarchical Encapsulation," results in a significant degree of reusability of the Elements and Behaviors created by an author.

As is illustrated below, authors can construct complex environments comprising a hierarchy of "Elements within Elements" interacting with one another. Within an Element, authors can create equally complex internal environments (together representing the Element's "personality") comprising a hierarchy of interrelated "Behaviors and other Modifiers within Behaviors."

Authors can selectively reuse these modular Element and Behavior object containers at virtually any level of complexity. Moreover, authors can "mix and match" Elements and Modifiers from differing environments at practically any level of the Structural and Behavioral Hierarchies. Furthermore, a team of authors can collaborate to create and test an entire application while individual authors refine their own modular, self-contained environments.

For example, multiple authors might collaborate to build a complex model of a "car" Element that contains one "engine," four "wheel" and various other Elements, as well as a "driving" Behavior that contains "steering," "braking" and various other Behaviors. Individual authors each might create one or more of these self-contained Elements and Behaviors, and then collaborate with one another to test all or a portion of the "car" and refine the desired interaction among the various Elements and Behaviors.

Moreover, a subsequent author working on a different application could reuse the entire "car" Element or merely one or more of the Elements or Behaviors contained therein. For example, the author might apply the "braking" Behavior to a "horse" or "airplane" Element in another application.

To create complex objects, authors need not create abstract "classes" or object templates that serve to relate objects to one another with respect to their characteristics. Elements, Behaviors and other Modifiers created by the author

are object instances. To permit Elements and Behaviors to "contain" other objects, the system links the Element and Behavior object instances to the other object instances contained within them (as is explained in greater detail below with respect to the system's implementation).

Thus, Elements and Behaviors are object containers--in this embodiment, object instances that can "contain" (i.e., be linked to) other object instances. To some extent, however, Elements do attain (at least temporarily) the characteristics they contain. Yet Elements merely provide an environmental frame of reference to their descendant Elements. Elements and Behaviors do not actually "inherit" the characteristics of the objects they contain.

Many embodiments of this "object container" relationship among objects are possible, including intersecting families of objects, multiple-parent relationships and various combinations of "one-to-many" and "many-to-many" child-parent relationships. In any event, by spawning object hierarchies, object containers provide the environments that facilitate their reusability via Hierarchical Encapsulation, discussed below.

Authors are thus free to create and work directly with objects that represent, at any level of complexity, the characters in their application and the individual characteristics "contained within" those characters, as well as the all-encompassing "environmental factors" that "contain" those characters and determine the nature of their interaction.

3. Object Hierarchies: Structural Hierarchy and Behavioral Hierarchy

When an Element contains another object within it, the Element object is called a parent and the other object is called its child. If the child object in turn contains another object, the child object is considered the parent of the object it contains, and so on.

This chain of parents and children is called an Element or (Structural) Hierarchy. In one embodiment, each parent can have multiple children, but each child has exactly one parent. Elements above and below a particular Element in the Structural Hierarchy can be referred to, respectively, as ancestors and descendants of that Element. Children of the same parent are called siblings.

Just as an Element can contain other Elements in the Structural Hierarchy, each Element also contains its own Modifier or Behavioral Hierarchy of Behaviors and "leaf" Modifiers (i.e. the Element's personality). These Modifiers modify the characteristics of the Element (e.g., by storing data in the Element or performing actions on behalf of the Element), often in response to messages from other Modifiers inside the same or another Element.

In the context of the Structural Hierarchy, an Element can be viewed as an environmental "frame of reference" for its descendant Elements. It is within that Element's environment that its descendant Elements exhibit their personalities. The Structural Hierarchy determines the manner in which those descendant Elements interact with one another within that Element's environment.

Similarly, the Behavioral Hierarchy determines the manner of interaction among an Element's internal Behaviors. In the context of the Element's overall personality, a Behavior also can be viewed as an environmental "frame of reference" for its descendant Behaviors and other Modifiers.

Continuing our "car" example, the "car" Element provides the environment in which the "engine" Element functions. If the car moves, the engine will move with the car automatically (as it would in a real car). Yet, the engine Element need not have any internal "movement" Behavior. By acting as the local coordinate system for its child Elements, the car becomes the engine's environmental frame of reference. This particular manner of isolating positional dependencies of Elements within the Structural Hierarchy, referred to as Hierarchical Relative Positioning, is discussed in greater detail below.

Similarly, the "driving" Behavior provides the environment in which the "braking" Behavior functions. If the driver does not step on the brakes, the "braking" Behavior will not be invoked--i.e., it will not receive a "brake pedal

depressed" message from its parent "driving" Behavior. By acting as the "gatekeeper" for messages intended for its child Behaviors, the parent "driving" Behavior becomes the "braking" Behavior's environmental frame of reference. This particular manner of isolating messaging dependencies of Elements and Behaviors (and other Modifiers) within the Structural and Behavioral Hierarchies, referred to as Hierarchical Message Broadcasting, is discussed in greater detail below.

Continuing the example, the "driving" Behavior must monitor the car's speed for a variety of reasons, as must the driver of a real car (e.g., to know when to release the brake). The speed of the car could be stored in a "car speed" Variable inside the Element, but not necessarily inside the "driving" Behavior because other Behaviors (e.g., an "Air Bag" Behavior detecting massive deceleration) may need to access this Variable. By making its child Variables accessible to its descendants, the car becomes the frame of reference for the "driving" Behavior. This particular manner of isolating data dependencies of Elements and Behaviors (and other Modifiers) within the Structural and Behavioral Hierarchies, referred to as Hierarchical Variable Scoping, also is discussed in greater detail below.

a. Types of Elements in the Structural Hierarchy.

As noted above, there exist two basic categories of Elements in this embodiment--Structural Elements and Media Elements.

Authors can utilize Structural Elements to group the contents of a title into organized sections, like the chapters of a book or the acts in a play. In this embodiment, the purely Structural Elements comprise Projects, Sections and Subsections.

Media Elements (including the Modifiers contained within them) comprise the primary content of an author's application. They typically are the characters or actors within a "Scene" (a special Media Element the primary purpose of which is structural--to contain other Media Elements). Any Media Element can be used structurally, as containers for other Media Elements, to form a more complex Element. Media Elements as their name suggests, typically are linked to raw media, such as text, sounds, pictures, movies or animations, in order to communicate with the user of the title.

(1) Projects.

A Project is a Structural Element that contains the entire title. Its children are Sections, which are described below. A Project also can contain Modifiers (including Behaviors) that store data or perform actions on behalf of the entire Project. The score of a game, for example, could be stored in the Project so as to be available to all other Elements and Modifiers.

(2) Sections.

A Section is a Structural Element that can be used by the author to organize logical portions of his title. It is most analogous to an act in a play or a chapter in a book. For example, an author of a U.S. travel game might create a Section for all Elements and Modifiers relating specifically to the state of California. Sections are parents to Subsections, as described below. As with Projects, Sections also can contain Modifiers. A "time-in-state" Behavior might, for example, track the amount of time the user spent in a particular state.

(3) Subsections.

A Subsection is a Structural Element that can be used by the author to further organize logical portions of a Section. Continuing the above example, the author of the travel game might create a Subsection for all Elements and Modifiers relating specifically to the San Francisco Bay Area. Subsections are parents to Scenes, and also can contain Modifiers.

(4) Scenes.

A Scene is a special Media Element that not only further organizes logical portions of a Subsection, but also contains other (non-Scene) Media Elements and Modifiers. Much like the scenes in a play, the Scene Element contains those

Media Elements and Modifiers the author deems necessary to impart to the user a portion of the interactive experience provided by the parent Subsection. Continuing our travel game example, one Scene might take place on the Pacific Coast, while another occurs further inland in the city of Palo Alto.

In one embodiment, there exists a special type of Scene Element known as a Shared Scene. A Shared Scene is used to organize Elements that will be visible across multiple Scenes, such as a background picture or an animated character that travels from Scene to Scene.

In a more refined embodiment, the author can share visible Elements across Subsections and Sections by having multiple Shared Scenes, or being able to select one or more Shared Scenes dynamically. In the latter case, rather than designating a Shared Scene statically at "authoring time," the author can, dynamically at "runtime," designate any Scene (or perhaps multiple Scenes) as a current Shared Scene.

(5) Media Elements.

Media Elements comprise the primary characters or actors in a Scene. They can contain other Elements (to form a more complex Element) and can be linked to raw media. A Media Element also can contain Modifiers, and thus Behaviors, which together constitute the Element's personality. As illustrated below, it is an Element's "dual hierarchy" that provides the environmental framework in which descendant Elements interact with one another and exhibit their own individual "personality."

b. Isolation of External Dependencies of Object Containers.

Elements and Behaviors are object containers that spawn hierarchies of descendant objects. Consider the sample illustration of the Structural and Behavioral Hierarchies in FIG. 1. When Media Elements are created, they are positioned automatically below the Scene in the Project's Structural Hierarchy. Thus, Element E1 108 is a child of Scene Sc1 106 in Subsection SS1 104 in Section S1 102 in Project P 101. Element E4 111 is a child of Element E1 108.

By isolating the external dependencies of the environment created by an Element or Behavior object container, the system provides for a significant degree of selective reusability of that Element or Behavior (or any object contained therein) in other environments, as is further demonstrated below. It is this isolation of an object container's external dependencies that allows the Elements and Behaviors created by an author to become an "environmental frame of reference" for their descendant Elements, Behaviors and other Modifiers, as was demonstrated in the "car" example discussed above.

In one embodiment of this system, three mechanisms are employed to isolate external dependencies within object containers: (i) Hierarchical Message Broadcasting--messages that are received by an Element or Behavior typically are "broadcast" down the Structural and Behavioral Hierarchies to its descendant Elements and Modifiers (though this effect can be overridden); (ii) Hierarchical Variable Scoping--Variables are accessible to all descendant Elements and Modifiers of that Variable's parent Element or Behavior; and (iii) Hierarchical Relative Positioning--an Element's position in the Scene is determined relative to the position of (i.e., using the local coordinate system of) its parent Element, and therefore changes as the position of its parent Element changes.

(1) Hierarchical Message Broadcasting.

A message sent to an Element or Behavior typically will be broadcast down the Structural and Behavioral Hierarchies. Elements pass down messages to their children, be they Modifiers or other Elements. Behavior pass down messages to their child Modifiers. The precise manner in which the various types of messages propagate down the Structural and Behavioral Hierarchies is discussed in greater detail below with respect to object messaging generally.

In one embodiment, the order in which messages are broadcast among Elements, Behaviors and other Modifiers at any given level of the Structural and Behavioral Hierarchies is determined by the order in which those objects appear in the "structure view window" (see, e.g., structure view window 330 in FIG. 4).

Initially, this is the order in which they are created. The author can modify this order simply by "dragging and dropping" or "cutting and pasting" the object's icon in the structure view window.

For example, referring again to FIG. 1, a message sent to Scene Sc1 106, under one embodiment, initially would be sent down to Element E1 108 (but not to any sibling Elements of Scene Sc1 106, such as Shared Scene Sc0 107). It would then be sent down to Element E4 111, and then to Element E2 109. From there it would be sent down to Modifiers M1 114, M2 115 and M3 116, and then to Element E3 110. It would then be sent down to Modifier M4 117, Behavior B1 118 and then down to Modifier M5 119. Finally, it would be sent to Behavior B2 120 and then down to Modifiers M6 122 and M7 123.

Note that, in this embodiment, Variables such as Variable V3 121 solely store data and thus do not respond to messages. Their data, however, can be read and written by other Modifiers, as for example, the Miniscript Modifier.

In this manner, a message sent to an Element or Behavior typically will be broadcast to all objects contained within that Element or Behavior. Upon receiving the message, those objects (Elements, Behaviors and other Modifiers) can then respond appropriately, performing whatever actions are specified by their current configuration.

By providing for the sending of messages to an Element or Behavior object container, and the broadcasting of those messages to the objects within that container, the system facilitates the isolation of messaging traffic to and from that container. In essence, the container becomes a higher-level abstract destination for messages intended not only for the container itself, but for "objects within the container."

Continuing the above example of the travel game, assume that Section S1 102 represents California, Section S2 103 represents Kansas and Subsection SS1 104 represents the San Francisco Bay Area. The author could create Behaviors to simulate the different weather patterns in these two states.

A "Midwest Weather" Behavior in Section S2 103 could simulate the weather in Kansas and periodically update a "temperature" Variable and send a "check temperature" message to that Section S2 103, which would be broadcast down the Structural and Behavioral Hierarchies to all Elements and Modifiers in each Scene in Kansas. This Behavior and the temperature Variable could both be contained at the Section S2 103 level of the Structural Hierarchy to simulate relatively constant weather throughout the state of Kansas.

Conversely, within California, Subsection SS1 104 (the San Francisco Bay Area) could contain two Scenes, one representing the Pacific Coast and the other representing Palo Alto. These Scenes could contain "Coast Weather" and "Inland Weather" Behaviors, respectively, each updating their own "temperature" Variable and sending a "check temperature" message to their respective Scenes. These separate Behaviors could simulate the differing weather patterns that occur even within a relatively small region of California.

"People" Elements in the various Scenes within the two states might be very complex, perhaps containing five or ten different Behaviors, including a Jogging, Swimming and a "Check Temperature" Behavior that check the temperature in response to a "check temperature" message. If the temperature deviates too far from the average temperature, the Check Temperature Behavior might send a "Very High Temperature" or "Very Low Temperature" message to its person Element, which will be broadcast down to the Swimming and Jogging Behaviors. The Swimming Behavior might be enabled by "Very High Temperatures" and disabled by "Very Low Temperature" and vice-versa for the Jogging Behavior.

By permitting the various "weather" Behaviors to broadcast their messages down the Structural and Behavioral Hierarchies (from the Section level in Kansas, and from the Scene level in California), the people Elements in either state will receive this message from their respective parent Scenes. The "weather" Behaviors need not target their message directly to a particular person Element, or even know which type of Element might respond to their "check temperature" message (e.g., person, animal, etc.).

The people Elements, therefore, could be moved from any Scene in Kansas to any other Scene in Kansas or California, and still respond appropriately to the temperature in that region. Even though the "check temperature" message is sent to a different environment or level of the Structural Hierarchy in Kansas (Section) than in California (Scene), it still reaches all relevant people Elements.

This degree of reusability of Elements and Behaviors is facilitated by the system's Hierarchical Message Broadcasting mechanism, which broadcasts messages sent to an object container to the objects it contains. This mechanism isolates the dependencies of a message at the abstract level of the environment represented by the object container to which that message is sent, as opposed to requiring the author to target the message directly to a particular "leaf" object.

As is discussed below, direct targeting of messages remains an option in this embodiment in the event that the author elects to forego a degree of reusability in favor of explicit target naming.

Hierarchical Message Broadcasting is used advantageously not only within but among Projects--Projects that might be located on the same computer, or on different computers on a Local Area Network or the Internet. Thus, if two Projects are constructed with substantially similar Structural Hierarchies (both topologically and nominally), then sending a Message from one Project to the other can be accomplished by the author in virtually the same manner as is sending a Message within the one Project, subject to the same rules of Message propagation upon arrival at the remote Project.

For example, consider two army platoons separated physically, but remaining in radio contact. Each platoon is led by a first lieutenant, followed by a master sergeant, who in turn leads lesser-grade sergeants and privates. The first lieutenant of the first platoon might send important messages directly to the first lieutenant of the second platoon, who would pass the message down the rank hierarchy as required. Less important messages might be sent directly between the master sergeants, and in some cases the first lieutenant of the first platoon might message the master sergeant of the second platoon directly (e.g., relieve your first lieutenant of his duties).

(2) Hierarchical Variable Scoping.

As demonstrated above, Elements and Behaviors are dependent upon messages to enable, disable and trigger certain actions, including those of their descendant Modifiers. Those Modifiers frequently are dependent upon Variables to perform actions on behalf of their parent Elements. Isolating Variable dependencies within Elements and Behaviors also facilitates reusability of those object containers.

The scope of Variables (data) accessible to a Modifier is another manifestation of that Modifier's environment. Variables are accessible to all descendant objects of that Variable's parent Element or Behavior. In other words, Variables are accessible within their parent's environment.

Hierarchical Variable Scoping serves to isolate an object container's dependencies on external Variables. By making Variables accessible to descendant objects of the Variable's parent, those descendant objects automatically "know about" those external Variables. A descendant Element or Behavior therefore becomes reusable in that the external Variables upon which it depends are isolated within ("known to") that Element or Behavior.

Returning to our travel game example, the "temperature" Variable in Kansas is accessible to the "Check Temperature" Behavior within any person Element of any Scene in Kansas. Because the "temperature" Variable is located at the Section S2 103 level, its descendant Elements include all Subsections therein, each Scene contained within those Subsections, each Element in each such Scene (including any people Elements), and finally the "Check Temperature" Behavior contained within such people Elements.

Moreover, if a person Element was moved from a Scene in Kansas to the Pacific Coast Scene in California, the "temperature" Variable in that Scene would be

accessible, automatically, to that person Element and to its "Check Temperature" Behavior. This degree of reusability is a direct result of the Hierarchical Variable Scoping mechanism.

In the above example, this mechanism isolates "temperature" Variable dependencies at the Section level in Kansas and at the Scene level within the San Francisco Bay Area in California. As noted above, the Hierarchical Message Broadcasting mechanism also isolates messaging dependencies (in particular the "check temperature" message) at those same levels of the Structural Hierarchy. Thus, in this example, these two mechanisms together make object containers below that level (such as the person Element) reusable across Sections of a Project.

(3) Hierarchical Relative Positioning.

Another environmental dependency of object containers, in this case limited to Elements, is the position of an Element relative to that of its ancestors. Rather than require the author to model the common circumstance in which the movement of one Element must be relative to the movement of another Element, the system, by default, determines an Element's position in the Scene relative to the position of (i.e., using the local coordinate system of) its parent Element.

In another embodiment, this effect could be made optional. In any event, a child Element can, as part of its own personality, move on its own initiative. Its position nevertheless remains relative to its parent Element. It therefore also will continue to move as its parent Element moves.

Thus, a child Element moves (changes position) within the Scene as its parent Element moves. This effect filters down the Structural Hierarchy below the Scene level. A common use of Hierarchical Relative Positioning is to model Elements physically contained within or attached to other Elements (e.g., a toy in a box or the wings of a bird). It also can be used to model Elements that, for one reason or another, tend to follow other Elements (e.g., planets orbiting the sun, or packs of animals that tend to move together).

Hierarchical Relative Positioning also serves to isolate an Element's dependencies on its environment, in that an author need not model this "follow my parent" movement. Upon reusing or reattaching an Element to another parent Element, the system automatically recalculates that Element's position relative to its new parent Element.

c. Selective Reusability through Adoption and Transplantation.

The selective reusability of object containers discussed above frequently takes one of two forms. If Elements are to be reused, they often will be placed in a new environment--i.e., given a new parent Element in the Structural Hierarchy. This process is referred to as "Adoption." In other words, the child Element has been "adopted" by a new parent Element.

The underlying implementation of Adoption is discussed in greater detail below. From the author's perspective, Adoption is a simple process at authoring time. For example, referring to FIG. 2, the author can break its current link and create a new link with the link tool 365 on the tool palette 36 with another Element in the layout view window 320. Alternatively, the author can drag and drop the Element's icon to the icon representing its new parent Element in the Structure Window 33.

Authors also may desire to move a Behavior or other Modifier from within its parent Element or Behavior to another parent Element or Behavior. This process is referred to as "Transplantation." In other words, a Behavior or other Modifier, and its associated capabilities, are "transplanted" to a new Element or Behavior, which then acquires those capabilities. Transplantation also can be accomplished quite easily at authoring time, by dragging and dropping the icon representing the Behavior or Modifier from its old parent to its new parent, either in the Layout Window 320 or in the Structure Window 33.

Adoption and Transplantation are quite similar, both from the author's perspective and in view of the manner in which they are implemented, as is

WEST

End of Result Set

Generate Collection

L2: Entry 2 of 2 File: USPT Mar 30, 1999

US-PAT-NO: 5890133
DOCUMENT-IDENTIFIER: US 5890133 A

TITLE: Method and apparatus for dynamic optimization of business processes managed by a computer system

DATE-ISSUED: March 30, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Ernst; Michael	Boeblingen	N/A	N/A	DEX

ASSIGNEE INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corp.	Armonk	NY	N/A	N/A	02

APPL-NO: 8/ 718055
DATE FILED: September 17, 1996

INT-CL: [6] G05B 13/00

US-CL-ISSUED: 705/7; 705/8, 707/2
US-CL-CURRENT: 705/7; 705/8, 707/2

FIELD-OF-SEARCH: 705/7, 705/8, 707/2, 707/102, 395/200.32

REF-CITED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5301320	April 1994	McAtee et al.	705/9
☐	5630069	May 1997	Flores et al.	705/7
☐	5721913	February 1998	Ackroff et al.	395/614

ART-UNIT: 274

PRIMARY-EXAMINER: Peeso; Thomas

ATTY-AGENT-FIRM: Clay; A. Bruce

ABSTRACT:

The invention relates to a method and a device for the dynamic optimization of business processes, the business process instances of a business process being

WEST

End of Result Set

Generate Collection

L1: Entry 2 of 2 File: USPT Apr 5, 1994

US-PAT-NO: 5301320
DOCUMENT-IDENTIFIER: US 5301320 A

TITLE: Workflow management and control system

DATE-ISSUED: April 5, 1994

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
McAtee; John D.	Shrewsbury	MA	N/A	N/A
Kennedy; Stephen M.	Hudson	MA	N/A	N/A
Piccolomini; Paul J.	Fitchburg	MA	N/A	N/A
Cerqua; Paul J.	Chelmsford	MA	N/A	N/A

ASSIGNEE INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Digital Equipment Corporation	Maynard	MA	N/A	N/A	02

APPL-NO: 7/ 722691
DATE FILED: June 28, 1991

INT-CL: [5] G06F 9/00, G06F 9/40

US-CL-ISSUED: 395/650; 364/DIG.1, 364/281.3, 364/281.8
US-CL-CURRENT: 705/9

FIELD-OF-SEARCH: 395/600, 364/281.3, 364/281.8, 364/DIG.1

REF-CITED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4356546	October 1982	Whiteside et al.	364/200
☐	4845739	July 1989	Katz	379/92

OTHER PUBLICATIONS

Milan Milenkovic, "Operating systems concepts and design", 1987, 111-116, McGraw Hill.

ART-UNIT: 236

PRIMARY-EXAMINER: Shaw; Gareth D.

ASSISTANT-EXAMINER: Katbab; A.

ATTY-AGENT-FIRM: Cesari and McKenna

ABSTRACT:

Methods and apparatus for defining, executing, monitoring and controlling the flow of business operations. A designer first defines a workflow by providing a template of business activities that expresses the manner in which these activities relate to one another. The system orchestrates performance of the tasks in accordance with the template; in so doing, it integrates various types of application software, and partitions tasks among various users and computers.

16 Claims, 5 Drawing figures
Exemplary Claim Number: 11
Number of Drawing Sheets: 5

BRIEF SUMMARY:

BACKGROUND OF THE INVENTION

A. Field of the Invention

The present invention relates to automated processing of business tasks and, more specifically, to a generic computer system that can be configured to define, execute, monitor and control the flow of business operations.

B. Description of the Related Art

The operation of a business can be viewed as an organized sequence of activities whose ultimate purpose is to produce a product or provide a service; this end result may be termed a "goal". Each business activity in the sequence involves performance of one or more items of work that bring the business one step closer to completion of the goal. Activities may be strictly ordered with respect to one another, conditionally ordered or completely unordered; they may be automated, partially automated or performed manually; and may be performed using resources within or outside the business.

In a large class of businesses, individual work items progress through the sequence of activities until they have been transformed into a finished item; that is, until the ultimate goal is achieved. Hereafter, such work items are referred to as "works in process", or WIPs, and the path followed by a WIP is termed a "workflow". For WIP-oriented businesses, resource and task management can prove difficult; this is particularly so where the WIP is processed according to a predetermined sequence of activities, each of which is carried out only if appropriate conditions are met, and by different personnel or machines using different pieces of information.

A useful example of such a business, and one to which we shall return later, is a mail-order enterprise. For purposes of this example, suppose that the business is divided into three departments: the front office, the warehouse floor, and the shipping dock. These departments execute the four main business activities: receiving and performing the initial check of an order; processing the order; performing final checks on the goods to be shipped; and actually shipping the goods. Such a scheme is illustrated in FIG. 1.

The initial order check consists of verifying the customer's credit rating, and determining whether the goods are available from inventory; these activities are performed sequentially. Processing the order consists of filling the order, an activity which involves the three parallel subactivities of sending a label, box and the merchandise to shipping; locating and packaging the ordered goods; and preparing an invoice, which involves the two sequential subactivities of generating an invoice and printing the invoice. The final-check activity involves inspection of the order, and is followed by the activity of shipping.

In this example, or processing path, can involve a number of separate resources: different personnel in each of the three departments, each group beyond initial intake requiring status information regarding the progress of the previous group before activity can be commenced; different computational resources (e.g., a third-party credit verification database, inventory-control equipment and spreadsheet and/or word-processor application programs); and

different criteria governing when various activities can begin.

Because of the variety of resources and the necessity of maintaining a relatively fixed business procedure, integrating these different resources can create dynamic logistical problems that interfere with efficient operation. Such problems arise from the need for communication among resources; the necessity of selectively combining resources from physically disparate locations (e.g., it may be necessary for the same operator to access different computer terminals to perform the credit check and then print a label); bottlenecks that reflect inefficiencies in resource allocation; and the need to segregate different activities for efficiency or security reasons (e.g., it may not be desirable for the same operator to print the label and also perform the credit check).

Unfortunately, while an ordinary computer system of sufficient capacity may be capable of executing most of the mail-order steps described above (using several application programs and with varying degrees of human and mechanical assistance), it is, at present, quite cumbersome to custom-tailor such a system to efficiently accommodate the specialized pattern of activities that defines the workflow; and even if such a "metasystem" can be configured, it is likely to be very difficult to modify should a change in workflow procedure become desirable.

DESCRIPTION OF THE INVENTION

A. Summary of the Invention

We have developed a generic computer system and architecture that can be straightforwardly configured to accommodate a user-defined workflow. A designer first defines a workflow by providing the system with a template of business activities that expresses the manner in which these activities relate to one another. Our system can integrate various types of application software, and is capable of partitioning tasks among various operators, computers and computer terminals as specified by the designer.

After the designer completes the template and the system is fully operational, users of the system, and the system itself, perform the various business tasks that define the work-flow in accordance with the workflow template. The configured system supervises and orchestrates performance of these tasks so that they occur in the specified order.

The system provides a new approach to the creation of large application systems by representing workflow tasks in a fully modular fashion. That is, an identifier for each task, along with a specification of the activities necessary for its completion and the manner in which the task relates to other tasks, are stored as a discrete, self-contained package. This allows the designer to alter the order and relationships among tasks without reconfiguring the entire workflow system, a considerable chore that would be necessary if the workflow were to be programmed in the traditional, linear fashion. Consequently, workflow design can proceed in an evolutionary fashion, with the designer altering the workflow as actual operations reveal bottlenecks or other inefficiencies.

The design of the system stems from our recognition that workflows can be defined by seven key relationships among activities:

1. Connectivity: describes the precedence relationship or the manner in which activities are interrelated and the order in which they occur.

2. Place: defines the place (e.g., the particular CPU on a network) at which a task is performed.

3. Timing: determines when activities begin, how long they persist, and when termination is appropriate.

4. Value: quantitatively describes the importance of a particular activity relative to other activities, thereby establishing priority relationships among activities.

5. Grouping: some activities can or must have some type of aggregative relationship, such as synchronization, sequential grouping or parallel grouping.

6. Scaling: describes the manner in which steps or activities can be decomposed into more granular tasks or recomposed into higher-level steps.

7. State: defines the status of a particular activity, WIP or of the workflow as a whole.

When installed in a computer, the programming of the present system resides, in a functional sense, above the computer's operating system (which controls basic hardware activities such as storage and retrieval of data) and below user-supplied application software, but is independent of both. System operation is driven by the occurrence of specific events that relate to the business process; these events can be "declared" by application software or otherwise recognized by the system during control operations.

The core system is organized into four components, which are implemented on and processed by a general-purpose computer: the Controller, the Controller Services Interface, the Manager Utility and the Manager Services Interface. Each of these systems is resident in permanent electronic storage facilities, and parts thereof are selectively retrieved and introduced into the volatile memory of the computer to facilitate processing. Although the system can be straightforwardly installed on a variety of computer systems, the primary criterion being sufficiency of processing power and speed to support operation of the four system components (although it is not necessary for all components to reside on a single machine) and accommodate the necessary application programs, the distributed nature of most workflow activity favors use of multi-user or network systems that allow both simultaneous processing and communication among user locations. Furthermore, because of the ability of the system to group execution of different activities, it is possible to use "parallel-processing" hardware for more efficient operation.

To configure the system for a particular application (after the components have been installed within the computer hardware), the designer first defines the workflow by decomposing the business process into an ordered description of discrete goals. Each goal can represent the completion of a business task or activity, or the occurrence of a specific business-related event. We define business goals that involve a single activity as "primitive" goals, and those specifying simultaneous or sequential execution of multiple activities as "compound" goals.

After constructing the workflow, the designer interacts with the Manager Utility (M/U) to create a workflow template. In operation, the M/U performs in a manner analogous both to word-processing and graphical editors; the designer establishes a workflow definition, against which work is ultimately processed, and sets system parameters. The M/U allows the designer to define the manner in which WIPs are created and identified, as well as the manner in which they will be used to control workflow operation.

The Controller supervises access to application programs, and assigns work to WIP-processing programs, written by the designer, hereafter referred to as "software agents", or SWAs. SWAs are scheduled by and executed under the control of the Controller and Controller Services Interface (CSI) components; SWAs can process WIPs directly, without communication with system users. The Controller is, in effect, the system's "engine"; as WIPs are created, the Controller processes them through the workflow activities as directed by the template (which was constructed by the designer using the M/U). After a WIP undergoes a template-specified task, the Controller proceeds to process the WIP according to the next task by identifying and activating the appropriate resource or resources.

More specifically, the SWA notifies the Controller, through the CSI, that processing of the WIP has been completed. Then, in accordance with restrictions contained in the template, the Controller identifies the next task or tasks (if any), locates the best available resource to process such task or tasks, and queues the WIP to that resource for processing. The CSI is a set of straightforward run-time routines and a server that facilitate communication

between the Controller, the SWAs and other application programs. The server acts as a front-end multiplexer for the Controller by receiving all communications directed thereto and selectively transmitting them to the Controller in accordance with availability and priority. The run-time routines establish links to SWAs and communication agents (discussed below) as appropriate during goal processing. The decision to separate communications from control operations represents a design choice; we have found that the increase in Controller processing efficiency achieved by eliminating its communication overhead justifies this separation.

The Controller can be apprised in two ways that a task has been completed. For automated tasks performed directly by SWAs, the CSI has invoked the executing program, so completion of processing of a WIP through a SWA suffices to signal conclusion of the corresponding task for that WIP. In other cases, however, the state of processing may not indicate the status of a task. For example, the task may be completed manually by business personnel who do not directly interact with any system component, or can involve the use of application programs over which the Controller does not have plenary command. In these situations, the event representing completion of the task is "declared" to the Controller, either by an application program (using a special command) or directly by the user through the M/U. Thus, in the above-described mail-order business, a fully automated credit check might be performed by a SWA that accesses a third-party database and evaluates the obtained data (in which case an event need not be declared), while order inspection is necessarily a manual activity requiring user intervention (and necessitating event declaration).

Event declarations are facilitated by programs hereafter referred to as "communication agents", or CMAs. These programs establish communication channels between the CSI and application programs or users, interact with users (either directly or through application programs) and allow users to notify the CSI of task completion.

After completion of a goal with respect to a particular WIP, the Controller selects the next goal or goals to be processed for the WIP based on considerations of connectivity, place, timing, value and/or class of WIP or resource.

DRAWING DESCRIPTION:

B. BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing discussion will be understood more readily from the following detailed description of the invention, when taken in conjunction with the accompanying drawings, in which:

FIG. 1 is a flow diagram illustrating an exemplary business structure of a mail-order enterprise;

FIG. 2 is an illustration of a representative workflow;

FIG. 3 depicts a processing loop representative of the manner in which a SWA iteratively processes WIPs;

FIG. 4 is a block diagram illustrating a representative system organization; and

FIG. 5 is a flowchart that illustrates overall operation of the process of the present invention.

DETAILED DESCRIPTION:

C. DETAILED DESCRIPTION OF THE INVENTION

1. Defining a Workflow

Before accessing the workflow system of the present invention, a designer first breaks down the sequence of business operations into a series of goals, each of

which represents a discrete business activity; each goal is then further decomposed into the tasks necessary for its accomplishment. FIG. 2 illustrates the manner in which the above-described mail-order enterprise can be represented as a series of goals. The first item 8 represents an event goal. Upon the occurrence of a triggering event (namely, entry of a new order) the Controller WIP through the workflow. (In practice, where large number of orders are to be expected, each WIP would most likely contain an ascending numerical component as a means of uniquely distinguishing the WIP.) In FIG. 2, the workflow to be processed upon declaration of Receive-Order (item 8) is represented as two compound goals 9 and 23 (called Process-Order and Final-Check, respectively) and the primitive goal 26.

The size of a compound goal (i.e., the number of component goals it specifies) is determined by the designer, and can stem from a variety of considerations. If a series of sequential tasks tend to be performed in association with one another, so that in altering the workflow the designer would be likely to shift, as a group, the relationship of these tasks to other tasks, convenience may dictate "packaging" them as a single compound goal consisting of multiple primitive goals. If multiple parallel tasks must all be completed before the next goal can become active, the parallel tasks are typically gathered into a single compound goal.

Thus, in FIG. 2, goals 8, 10, 12, 14 are both primitive and sequential; a goal that sequentially follows a previous goal is said to "depend" on that previous goal (so that goal 10 depends on goal 8, goal 12 on goal 10, and goal 14 on goal 12).

The three goals 18a, 18b, 18c collectively form a compound goal 18 that depends on goal 14, while the next goal 20 depends on the compound goal 18, and the next goal 22 on goal 20. Goals 10, 12, 14, 18, 20 and 22 collectively form compound goal 9, Process-Order. Goals 24 and 25 are primitive and dependent, and collectively form a compound goal 23, Final-Check.

The designer develops this workflow using the M/U, which operates like an editing facility. The function of the M/U is to generate a database of workflow descriptions that includes goals, relationships among goals and characteristics associated with each goal. These are interpreted and processed by the Controller during system operation to implement and enforce the workflow.

The M/U interacts with the user to obtain certain key pieces of information relating to each goal. This information is entered into fields that define a data structure for that goal, and the set of data structures for all goals is organized into a table. This table is referred to as the "workflow definition database". The fields for each goal include:

GOAL NAME: An alphanumeric string uniquely identifying the goal.

TYPE OF GOAL: As more fully described below, a goal can be categorized as a "task" goal, an "event" goal, a "group" goal or a "sequence" goal.

RELATED GOALS: If the goal is a compound goal, the names of all other goals that define the compound goal (which may itself contain compound goals).

TRIGGER GOAL: If the goal is an event goal, the name of the goal triggered by occurrence of the event.

NODES: Lists the system CPUs that are operative during the goal activity. This field allows the Controller to maintain control over workflow resources by allowing the goal to be processed only on specified CPUs. If no data is entered into this field, the system imposes no node restrictions.

PRIORITY: Goals operating in parallel can make overlapping demands on system resources. The priority factor provides data with which the Controller can allocate resources according to the importance of accomplishing a particular goal. In the preferred embodiment, priority is indicated by a numerical weight ranging from 1 to 32.

STATUS: A goal can be ENABLED or DISABLED. In the latter case, workflow execution is suspended until the goal is again ENABLED. A goal is typically

DISABLED if execution of the related SWA results in an error condition.

TIME QUALIFIERS: A goal can be restricted to running (i.e., remain "valid") only during particular times during the day, and can be given a maximum allowed execution time.

The designer enters data into each of the set of fields corresponding to each goal; the graphical appearance of the prompts and the nature of the designer's interaction with the M/U are not critical, and a wide variety of suitable approaches are readily implemented by those skilled in the art. In our preferred embodiment, the M/U offers both a graphical mode and a command-driven mode, allowing the designer to enter goals as a series of boxes that resemble the configuration shown in FIG. 2, or as a sequence of commands. Regardless of the details of data entry, the pattern of the designer's interaction with the M/U initially involves entry of goals and goal information into the workflow definition database. Thereafter, the designer creates the various computer programs (i.e., SWAs and CMAs) that facilitate or actually carry out these tasks.

A "task" goal is one that is fully executed by a SWA within the system. An "event" goal relates to an activity that occurs outside the system, the completion or which is made known to the system by a CMA, which may trigger processing of another, specified goal. Thus, item 8 denotes an event goal which, upon the occurrence of an event, initiates processing by the system. That event represents entry of new order information by a human operator, which apprises the system that an order has been received. More specifically, the designer has written a CMA that awaits a request for order entry by an operator and, upon receipt of such a request, causes the Controller to create a WIP (according to steps discussed in further detail below) and execute of subsequent goals.

The first goal entered, Process-Order, is a compound "sequence" goal that represents a large portion of the workflow, and which will contain subordinate goals that further define the workflow activities. The RELATED GOALS field for Process-Order would contain the names (at the next-highest level of generality) of all the goals shown in FIG. 2 as being within Process-Order.

Goal 10 involves entry of customer information into, for example, a commercial or custom-designed data-entry program. As discussed more fully below, the user enters the data and, upon completion of this activity, the CMA declares an event to the CSI indicating that the data has been entered. Thus, this goal represents an event goal, because the system is apprised of the completion of information entry. If goal 12 involves a fully automated credit check (as is common in many mail-order businesses), this step is represented as a task goal, and is fully implemented by a SWA. Goal 14 requires human activity (assuming a physical search to be necessary), and is therefore an event goal. Once again assuming the need for physical transfers and acknowledgments of arrival in the shipping department, goals 18a, 18b and 18c are event goals that make up the compound definition of goal 18.

Partial data structures for goals 8, 10, 12, 14, and 18 during the time the operator keys in the order information (for purposes of this example, on CPU 1 of the system network) are shown in the following table:

TABLE 1 _____ Goal Goal Goal 8 10 12 Goal 14
Goal 18 _____ GOAL Receive- Enter- Verify-
Determine- Prepare- NAME Order Order Credit- Whether- Order Rating Goods-In-
Inventory TYPE Event Event Task Event Event OF GOAL RE- -- -- -- -- Prepare/
LATED Send- GOALS Label, Send- Box, Send- Merchandise TRIG- Process- -- -- --
-- GER Order GOAL CODES CPU1 CPU1 Any Any Any

WIPs are also described by data structures containing information fields; these are generated by the Controller as WIPs are created, and stored in a database. The fields for each WIP include:

WIP IDENTIFIER: This is assigned by the Controller (in accordance with instructions issued by a CMA or SWA), or specified by the user when the WIP is created.

TRIGGERING EVENT: The event which, when declared, results in creation of the WIP and initiation of goal processing for that WIP.

TRIGGER GOAL: Some, though not all, event goals have associated trigger goals. If the goal following an event goal lies at the same hierarchical level as the event goal (e.g., it is a primitive goal that follows a primitive event goal in sequence), no trigger goal is stated. On the other hand, if completion of the event goal results in the initiation of new processing at a different hierarchical level (e.g., processing of a new compound goal), the next goal is entered as the trigger goal. More specifically, the trigger goal is the first goal processed as a result of the triggering event, expressed at the highest level of generality (i.e., if the first goal is part of a compound goal, this field contains the name of the compound goal).

WIP PRIORITY: For purposes of queuing WIPs to a SWA, each WIP is assigned a priority factor to indicate its importance relative to other WIPs. As with goal priorities, the weighting factor ranges from 1 to 32.

WIP-GOAL STATUS: A WIP can be ACTIVE, INACTIVE or COMPLETE with respect to a particular being processed for the WIP; accordingly, each of these designations is applied to a WIP on a per-goal basis (unlike the other fields, which apply regardless of which goal or goals are currently active for the WIP). The status conditions are used to respond to user or system inquiries regarding the WIP. The COMPLETE condition alerts an inquirer that the WIP has been fully processed through the goal that was the subject of the inquiry; the ACTIVE condition indicates that the WIP is currently being processed by the subject goal, and the INACTIVE condition indicates that the WIP has not yet been processed through this goal.

WIP STATUS: Independent of any particular goal, the WIP itself can be ACTIVE, IDLE or PAUSED. ACTIVE WIPs are those being processed at the time of the status inquiry; an IDLE condition indicates that the WIP has been fully processed, and the PAUSE condition indicates that processing has been interrupted manually.

TEMPLATE: Each WIP contains a copy of the goal-processing template relevant to that WIP. To reduce the necessary storage space, each goal contained in the WIP's template representation can be assigned a unique identification sequence or numeral, and goal information retrieved as necessary from the goal database during WIP processing.

Each WIP is generally associated with some form of application data, which can be handled in one of two ways. In the simpler embodiment, the workflow system is never exposed to the underlying WIP data. If processing of a goal requires retrieval and manipulation of the data, this is performed by the user and its completion signaled by declaration of an event. In the more complex embodiment, the WIP identifier is used to designate a data file that stores WIP data in whatever form the designer specifies. This embodiment typically requires more complex SWAs and CMAs to address and perform operations on the data.

For example, in the more complex embodiment, the WIP New-Order-Received can designate a blank business form that is represented and stored electronically. As the WIP moves through the various goals, information is entered by users (e.g., order information during activity 10) and the system (e.g., after completion of a fully automated credit check during activity 12, the form is marked to show the results).

The goal and WIP status fields allow users to query the system regarding the status of a particular activity or WIP. These inquiries are handled by the Manager Services Interface (MSI), which sorts through the relevant database fields. For status inquiries pertaining to an activity. The MSI retrieves and returns to the inquiring user the identities of all WIPs queued to that activity and the status of each such WIP; if the user's inquiry relates only to a particular WIP, the system indicates either (a) whether the WIP is ACTIVE, IDLE or PAUSED; or (b) with respect to a particular goal, whether the WIP is ACTIVE, INACTIVE or COMPLETE as to that goal.

2. Operation of the Controller and Controller Interface

As described above, the workflow is decomposed into a series of goals, and each business goal is decomposed into a series of component goals for processing. The Controller initiates and supervises processing of goals by executing or enabling use of the optimal (or available) resources (which may be human or computational). The performance or a goal-related task is initiated at the completion of a previous task, or in response to the occurrence of an event.

For each WIP, the relevant tasks (i.e., those associated with the goals listed in the WIP's data structure) are performed in sequence. Non-repetitive, sequential processing of tasks is facilitated by a logical pointer (or pointers) to the currently operative goal (or goals) in the template contained in the WIP data structure. After a SWA has finished executing a goal and the WIP has entered the COMPLETE state for that goal, or when an event indicating completion of the goal is declared, the system performs operations collectively referred to as "goal completion". These operations include advancing the pointer and evaluating the current state of processing. If the completed goal is the last component of a compound goal, the compound goal is itself considered complete.

After goal completion, the Controller determines whether further processing is necessary. If the next goal lies at a hierarchical level different from that of the just-completed goal, the Controller inspects the database to identify the next primitive item of processing. This iterative, nested process is referred to as "goal activation". Because it is necessary to refer to the goal database in order to determine the task or tasks associated with each goal (as well as the component goals associated with a designated compound goal), the system's database organization is preferably relational, allowing the logical pointer to specify goals at arbitrary levels of granularity. In the preferred embodiment, database-access programming is written in standard SQL code, allowing straightforward interface to relational database products such as RDB, Oracle and Ingres.

Since the system is organized as a database, coincident workflow executions can be accommodated in assembly-line fashion; that is, multiple goals can be simultaneously active for different WIPS (although a single CPU can process only one goal at a time). The order in which WIPs are processed for a particular goal depends on the chosen hardware implementation and may also depend on the priority associated with the WIP, as discussed below. If, for example, the system is configured to operate with parallel-processing or distributed architectures, multiple CPUs can simultaneously execute a particular SWA for different WIPs, enhancing the overall processing speed of multiple workflows commensurately.

Ordinarily, performance of any task requires execution of a SWA or a CMA; even if actual processing of the task is performed by a user outside the system environment (e.g., manually or through use of an application program that communicates with the Controller), a CMA initiates communication with the user and issues a confirmation to the Controller (via the CSI) that the task is complete.

As discussed in greater detail below, SWAs are invoked and executed by the Controller (via the CSI, which also interfaces to CMAs) to accomplish the specific tasks that lead to completion of a goal. Ordinarily, the SWA associated with a particular goal is identified by the name of that goal, enabling the CSI easily to access and activate the SWA at the appropriate point in the workflow. SWAs process WIPs, and the results of this processing can include modification of the WIP and/or production of a new WIP, in which case it is given a separate WIP identifier as discussed above. As explained previously, the workflow is executed on individual WIPs, and is repeated as necessary (and in the manner allowed by the hardware configuration) to process multiple WIPs. The way in which a particular SWA iteratively processes WIPs is illustrated by the processing loop shown in FIG. 3.

When the rate at which WIPs are available for processing by a particular SWA exceeds the rate at which the SWA can process an individual WIP, the WIPs must be queued. Queuing is managed by the Controller, and processing is most straightforwardly performed on a FIFO basis; however, WIP priority values can also be used to determine the order in which queued WIPs are processed.

With respect to a particular WIP, only SWAs relating to the current goal or goals are allowed to process the WIP. SWAs and system resources belonging to other goals remain unused for that WIP (e.g., users cannot process the WIP on system CPUs or application programs belonging to goals other than the current goal). A goal is considered completed for a particular WIP when the associated SWA has fully performed the automated processing representative of the task; completion of goals and triggering of new goals will be discussed below.

SWAs can be written in any computer language, so long as they are capable of complying with certain conventions associated with the present system. These conventions consist of a set of status messages that are generated by the SWA and communicated to the CSI to allow the Controller to supervise processing. The CSI establishes a communication channel or link between itself and the SWA upon issuance of a request by an activated SWA, which can reside at any logical system location. It then proceeds to execute the SWA according to the processing loop shown in FIG. 3.

The status messages generated by the SWAs include:

CSI$SWA.sub.-- OPEN.sub.-- CTRLR.sub.-- COMM: This message is issued by the SWA as a communication request to the CSI. A response by the CSI, which the SWA is configured properly to receive, indicates that communication has successfully been established.

CSI$SWA.sub.-- RECEIVE.sub.-- COMMAND: This message indicates successful initialization of the SWA, and that the CSI can now issue commands and data (in the form of WIP identifiers) to the SWA for processing. The CSI responds by providing commands and WIP identifiers to the SWA.

CSI$SWA.sub.-- START.sub.-- WIP.sub.-- PROCESSING: This message indicates the onset of processing by the SWA.

CSI$SWA.sub.-- END.sub.-- WIP.sub.-- PROCESSING: This message indicates that the WIP has been processed.

CSI$SWA.sub.-- RELEASE.sub.-- COMMAND: This message indicates to the Controller (via the CSI) that the current command is no longer being processed.

CSI$SWA.sub.-- CLOSE.sub.-- CTRLR.sub.-- COMM: This message closes communication between the CSI and the SWA.

CSI$SWA.sub.-- REQUEST.sub.-- GOAL.sub.-- PROCESSING(argument): This message requests the Controller to process the goal identified in the argument. It can be used to facilitate conditional branching, as discussed below.

The CSI uses these messages to monitor the progress of the SWA as it traverses the processing loop shown in FIG. 3. The foregoing messages are interpreted by the CSI during this loop as follows:

TABLE 2

SWA Message	Description
##STR1##	The SWA process has been invoked, has performed any necessary initializations, and now establishes communication with the Controller.
##STR2##	The SWA receives a processing command from the CSI.
##STR3##	The SWA enters this state as soon as it is ready to begin processing.
##STR4##	The SWA has completed the assigned processing. .
##STR5##	All "clean-up" operations have been performed, and the SWA is available to process another WIP.

As shown in Table 2, after completing one WIP assignment, the SWA notifies the CSI that it is available for a new assignment before looping back up to receive a new processing command. When processing of a WIP is completed, as reflected by the CSI$SWA.sub.-- END.sub.-- WIP.sub.-- PROCESSING message, the Controller performs goal completion.

However, not all tasks can be fully executed internally using SWAs. If, for

example, user action is necessary to complete a goal, the goal is an "event" goal and progress through the workflow depends on declaration of the appropriate event. In such a case, the user interacts in a manner appropriate for the application to indicate that work associated with the event has been completed. At this point, the CMA declares an event to the Controller, which performs goal completion.

Usually, CMAs (rather than SWAs) create WIPs, and are programmed to create or obtain the WIP identifier and priority. The CSI is apprised by the CMA that a WIP is being created, and the CSI then passes the information to the Controller, enabling the Controller to create a data structure for the WIP; the CSI obtains the WIP and priority from the CMA, and generates the remaining WIP information based on the currently active workflow template. All of this information is entered into the new WIP data structure.

Communication between the CSI and the CMA is facilitated by another set of message-passing routines, some of which are analogous to the SWA messages, and include:

CSI$OPEN.sub.-- CTRLR.sub.-- COMM: This message is issued by the CMA as a communication request to the CSI. A response by the CSI, which the CMA is configured properly to receive, indicates that communication has successfully been established.

CSI$DECLARE.sub.-- EVENT(argument): This declaration indicates to the CSI that the work associated with the event has been completed. The argument contains the name of the processed event goal and WIP.

CSI$CLOSE.sub.-- CTRLR.sub.-- COMM: This message closes communication between the CSI and the CMA.

The interaction between the Controller and an external activity is illustrated as follows. In FIG. 2, entry of new order information results in declaration of the event Receive-Order and creation of the WIP New-Order-Received. Completion of goal 8, indicated by the event declaration, initiates execution of the compound goal Process-Order. The Controller evaluates which goal or goals may be activated as a result of the event declaration, advances the WIP template pointer to the first primitive goal 10 (Enter Order), and proceeds to execute this goal. From the system's viewpoint, goal 10 consists of facilitating the user's entry of data and declaring the event Enter-Order to inform the Controller that goal 10 has been achieved.

In actual operation, a pre-programmed CMA enables the user's CPU and initializes the data-entry program at that CPU. The user then interacts directly with the data-entry program, without communication to the workflow system. When the user signs off, the CMA issues a CSI$DECLARE.sub.-- EVENT message to the CSI, where the argument is Enter-Order (the event representing the work just completed by the user). When the Controller receives this message via the CSI, it updates the WIP, and proceeds to perform goal completion.

Once again, the degree of control the system exerts over user data (i.e., the customer information entered via the data-entry program) represents a design choice. System operation can proceed in a manner involving no interaction with the user's data. In this case, updating the WIP consists essentially of advancing its template pointer and revising its status; if customer information is to be modified or other data generated during the course of the workflow, users are responsible for data access and entry. If on the other hand, system is to manage user data, the CMAs contain file-management capability so that data can be brought directly to the attention of the user or modified internally.

Activity processing is obviously simpler if all tasks within a compound goal are self-executing, and all operations occur internally within the workflow system. This can be true of activity 12 in FIG. 1, which, in our example, consists of the following tasks: accessing and establishing communication with a third-party credit-rating database; providing appropriate information to the database for evaluation; receiving credit information from the database; interpreting the data; and, if the customer's credit is satisfactory, generating a CSI$SWA.sub.-- END.sub.-- WIP.sub.-- PROCESSING message. Receipt

of this message by the Controller is interpreted as completion of goal 12. All of these tasks can be performed directly by one or more SWAs and CMAs; however, if the system does not access user data, the user provides the necessary customer information directly.

If the customer fails to meet the minimum credit requirements, the system can be configured to respond in any of several ways. One approach is to provide for conditional processing within the structure of the workflow, using the CSI$SWA.sub.-- REQUEST.sub.-- GOAL.sub.-- PROCESSING(argument) message, thereby causing transfer of goal execution from one sequence of goals to another (which can itself lead back into the initial sequence via a second CSI$SWA.sub.-- REQUEST.sub.-- GOAL.sub.-- PROCESSING(argument) message).

The system's database organization allows use of the invention in conjunction with a number of different hardware and software configurations. In its simplest form, the system is utilized with a single processor having one or more terminals. As each goal is completed, processing proceeds to the next activity. If the hardware system offers multitasking capability, processor resources can be partitioned to accommodate execution of multiple workflows in different states of processing. The present invention can also be configured to operate with distributed or parallel-processing architectures. For example, individual processors in a multiprocessor system can each be allocated to a single goal, with the Controller (itself run by a single processor) disseminating WIPs to the different system components for concurrent processing.

Such a system is illustrated in FIG. 4, which shows the manner in which system components interact. The system contains a series of CPUs denoted generically by reference numerals 30 and 32; one or more mass-storage devices 34, which store data files, operating systems and all workflow-system components; a workstation 36, which is a separate CPU (or a set of parallel CPUs) responsible for executing and managing all system components; and the server's main volatile memory 38, which contains all workflow-system components during operation as well as the operating system necessary for hardware operation. The components appearing in server memory 38 are copied therein from mass-storage devices 34 during initialization and processing, as appropriate, under the direction of server 36.

During system operation, users on the CPUs interact with application programs or the MSI as goals are processed. Server 36 executes SWAs and responds to event declarations until each WIP has been processed through all of the goals contained in its template, as well as any goals added thereto by conditional branching.

Refer, finally, to FIG. 5, which illustrates overall operation of the process of the present invention. At step 50, SWAs and CMAs are initialized. At step 52 data specifying execution parameters associated with each SWA and CMA is stored in system memory. Operation begins with analysis of the execution parameters at step 54, from which an execution sequence, specifying the order in which the SWAs and CMAs are to be executed, is derived in step 56. At this point the system is ready to receive WIP items (ordinarily, directly from a user) and store them in system memory, as indicated at step 58. Finally, at step 60, the SWAs and CMAs are executed according to the execution sequence, thereby implementing the user's workflow.

The terms and expressions which have been employed are used as terms of description and not of limitation, and there is no intention, in the use of such terms and expressions, of excluding any equivalents of the features shown and described or portions thereof, but it is recognized that various modifications are possible within the scope of the invention claimed.

CLAIMS:

What is claimed is:

1. An apparatus for defining, executing, monitoring and controlling the flow of business operations, the apparatus comprising:

a. at least one interface means for receiving information from a user;

b. storage means, coupled to the interface means, for electronically storing user-provided data representative of a work item;

c. instruction storage means for electronically storing a plurality of execution modules, each containing instructions that facilitate performance of a specified goal representing a business operation;

d. database storage means, coupled to the interface means, for storing a list of the execution modules that includes, with respect to each module, data specifying execution parameters associated with each module and an importance level associated with each parameter, said execution parameters related to at least one of (i) previous execution of at least one other specified module, (ii) specified place of module execution, (iii) specified duration of module processing, (iv) a specified time during which module execution may take place, and (v) a priority value associated with the module, wherein such execution parameters may be modified by a user via an interface means without disturbing the execution parameters associated with other modules;

e. at least one processor means capable of executing the stored instructions; and

f. control means, coupled to the processor means and the storage means, for analyzing the execution parameters associated with the execution modules and, based on the importance levels, deriving therefrom an execution sequence for the modules, and directing the processor to sequentially execute the instructions contained in each module in the order specified by the execution sequence and on the work-item data.

2. The apparatus of claim 1 wherein execution by the processor means of at least some of the modules includes performance of specified operations by business personnel, and further comprising communication means, coupled to at least one interface means and the processor means, for receiving, via an interface means, notification of completion of the specified operations.

3. The apparatus of claim 1 wherein the processor means is further adapted for electronically associating a work item with a unique alphanumeric identifier, the processor and control means being adapted for simultaneous processing of multiple work items.

4. The apparatus of claim 3 wherein at any particular time, each module is executed with respect to a single identified work item.

5. The apparatus of claim 3 wherein at any particular time, each module can be executed with respect to a plurality of identified work items.

6. The apparatus of claim 1 wherein the parameters can be defined so as to require execution of a plurality of modules as a condition of executing a subsequent module.

7. The apparatus of claim 2, wherein the parameters include specification of at least one specified place at which a task required for execution of a module is performed.

8. The apparatus of claim 1 wherein the parameters include specification of the duration of processing attending execution of a module and specified times during which such execution can take place.

9. The apparatus of claim 6 wherein the parameters include a priority value associated with each module.

10. The apparatus of claim 1 further comprising means, coupled to at least one interface means, for enabling a user to modify the parameters associated with each module, thereby altering the order in which the modules are executed by the processor means.

11. A method for defining, executing, monitoring and controlling the flow of business operations represented by a plurality of electronically stored

execution modules, each of which facilitates the performance of a specified goal representing a business operation, the method comprising:

a. receiving, via an interface means, electronically encoded data, representative of a work item, from a user;

b. electronically storing the work-item data in a first computer memory;

c. creating and electronically storing, in a second computer memory, a list of the execution modules that includes, with respect to each module, data specifying execution parameters associated with each module and importance level associated with each parameter, said execution parameters relating to at least one of (i) previous execution of at least one other specified module, (ii) specified place of module execution, (iii) specified duration of module processing, (iv) a specified time during which module execution may take place, and (v) a priority value associated with the module, wherein such execution parameters may be modified by a user via an interface means without disturbing the execution parameters associated with other modules;

d. analyzing the execution parameters associated with the execution modules and, based on the importance levels, deriving therefrom an execution sequence for the modules; and

e. electronically executing, by means of a computer processor, the instructions contained in each module in the order specified by the execution sequence on the work-item data.

12. The method of claim 11 wherein execution of at least some of the modules includes performance of specified operations by business personnel, and further comprising the step of electronically communicating, in accordance with the stored instructions, with business personnel to perform or obtain notification of completion of at least some of the specified business operations.

13. The method of claim 11 wherein the parameters can specify that execution of a plurality of modules is a condition of executing a subsequent module.

14. The method of claim 12 wherein the parameters can specify at least one specific place at which a task required for execution of a module may be performed.

15. The method of claim 11 wherein the parameters can specify the duration of processing attending execution of a module and designated times during which such execution can take place.

16. The method of claim 11 wherein the parameters can specify a priority value.

File 351:DERWENT WPI 1963-1999/[illegible]=, UM=, & UP=199954
(c) 1999 DERWENT INFO LTD
File 347:JAPIO OCT 1976-1999/SEP(UPDATED 991229)
(c) 1999 JPO & JAPIO
File 344:Chinese Patents ABS Apr 1985-1999/Dec
(c) 1999 European Patent Office

Set	Items	Description
S1	1	AU="BOWMAN-AMUAH M"

MUAH M"
?t s1/7

1/7/1 (Item 1 from file: 351)
DIALOG(R)File 351:DERWENT WPI

011235330 **Image available**
WPI Acc No: 97-213233/199719
Data communications network for computer communications with different operating system - has local area network with coupled wireless local loop, and packet switched network coupled to loop, where loop conforms to transmission control protocol internet control transmission and addressing protocol
Patent Assignee: INTERTRADE COMPUTER CONSULTANTS INC (INTE-N)
Inventor: *BOWMAN-AMUAH M*
Number of Countries: 070 Number of Patents: 002
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Main IPC	Week
WO 9712456	A1	19970403	WO 96US15598	A	19960924	H04J-003/02	199719 B
AU 9673794	A	19970417	AU 9673794	A	19960924	H04J-003/02	199732

Priority Applications (No Type Date): US 95533709 A 19950926
Cited Patents: US 5347516; US 5351237; US 5379290; US 5410754; US 5463623; US 5517620; US 5519705; US 5521914
Patent Details:

Patent	Kind	Lan	Pg	Filing Notes	Application	Patent
WO 9712456	A1	E	19			

Designated States (National): AL AM AT AU AZ BB BG BR BY CA CH CN CZ DE DK EE ES FI GB GE HU IL IS JP KE KG KP KR KZ LK LR LS LT LU LV MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK TJ TM TR TT UA UG UZ VN
Designated States (Regional): AT BE CH DE DK EA ES FI FR GB GR IE IT KE LS LU MC MW NL OA PT SD SE SZ UG

AU 9673794	A		Based on		WO 9712456	

Abstract (Basic): WO 9712456 A
The data communications network includes a local area network (LAN) (32), and a wireless local loop (WLL) (34) coupled to the LAN. A packet switched network (36) is coupled to the wireless local loop. This loop conforms to a transmission control protocol internet control transmission and addressing protocol (TCP/IP).
The packet network conforms to a TCP/IP transmission and addressing protocol, or to a UDP transmission and addressing protocol. The wireless local loop conforms to an ethernet protocol, and is a wireless T1/E1 channel with a coder decoder connected to it.
ADVANTAGE - Provides low cost and flexible system for connecting widely separated local area network or individual computers.
Dwg.2/8
Derwent Class: T01; W01
International Patent Class (Main): H04J-003/02
?

File 348:EUROPEAN PATENTS 1978-99/DEC W52
(c) 2000 EUROPEAN PATENT OFFICE

Set	Items	Description
S1	0	AU=(BOWMAN-AMUAH M? OR BOWMAN AMUAH M?)

File 123:CLAIMS(R)/CURRENT LEGAL STATUS 1980-1999/DEC 28
(c) 2000 IFI/CLAIMS
File 340:CLAIMS(R)/US PATENT 1950-99/DEC 28
(c) 1999 IFI/CLAIMS(R)

Set	Items	Description
S1	0	AU=(BOWMAN-AMUAH M? OR BOWMAN AMUAH, M?)

```
File   2:INSPEC 1969-1999/Nov
         (c) 1999 Institution of Electrical Engineers
File   8:Ei Compendex(R) 1970-1999/Nov W4
         (c) 1999 Engineering Info. Inc.
File   6:NTIS 64-1999/JAN W4
         COMP&DISTR 1998 NTIS, INTL COPYRIGHT ALL RIGH
File  99:Wilson Appl. Sci & Tech Abs 1983-1999/Nov
         (c) 1999 The HW Wilson Co.
File 144:PASCAL 1973-2000/DEC
         (c) 2000 INIST/CNRS
File  77:CONFERENCE PAPERS INDEX 1973-2000/JAN
         (c) 2000 CAMBRIDGE SCI ABS
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec
         (c) 1998 Inst for Sci Info
File  34:SCISEARCH(R) CITED REF SCI 1990-2000/JAN W1
         (c) 2000 INST FOR SCI INFO
File 108:AEROSPACE DATABASE 1962-1999/DEC
         (c) 1999 AIAA
File 233:Microcomputer Abstracts 1981-1999/Dec
         (c) 1999 Information Today Incl.
File 238:ABS. IN NEW TECH & ENG. 1981-2000/DEC
         (c) 2000 REED-ELSEVIER (UK) LTD.
File  65:Inside Conferences 1993-1999/Jun W3
         (c) 1999 BLDSC all rts. reserv.
File  94:JICST-EPLUS 1985-2000/SEP W4
         (c)2000 JAPAN SCIENCE AND TECH CORP(JST)
File  62:SPIN(R) 1975-1999/Nov W3
         (c) 1999 American Institute of Physics
File  35:Dissertation Abstracts Online 1861-1999/Oct
         (c) 1999 UMI

Set     Items   Description
S1          0   AU=(BOWMAN-AMUAH, M? OR BOWMAN-AMUAH M? OR BOWMAN AMUAH, M?
              OR BOWMAN AMUAH M?)
```

File 275:GALE GROUP COMPUTER DB(TM) 1983-2000/JAN 07
(c) 2000 THE GALE GROUP
File 674:COMPUTER NEWS FULLTEXT 1989-1999/DEC W2
(c) 1999 IDG COMMUNICATIONS
File 16:GALE GROUP PROMT(R) 1990-2000/JAN 07
(c) 2000 THE GALE GROUP
File 15:ABI/INFORM(R) 1971-1999/Dec 15
(c) 1999 Bell & Howell
File 98:General Sci Abs/Full-Text 1984-1999/Oct
(c) 1999 The HW Wilson Co.
File 148:GALE GROUP TRADE & INDUSTRY DB 1976-2000/JAN 07
(c)2000 THE GALE GROUP
File 636:GALE GROUP NEWSLETTER DB(TM) 1987-2000/JAN 07
(c) 2000 THE GALE GROUP
File 624:MCGRAW-HILL PUBLICATIONS 1985-2000/JAN 06
(c) 2000 MCGRAW-HILL CO. INC
File 9:BUSINESS & INDUSTRY(R) JUL/1994-2000/JAN 07
(c) 2000 RESP. DB SVCS.
File 88:GALE GROUP BUSINESS A.R.T.S. 1976-2000/JAN 07
(c) 2000 THE GALE GROUP
File 47:GALE GROUP MAGAZINE DB(TM) 1959-2000/JAN 07
(c) 2000 THE GALE GROUP
File 370:Science 1996-1999/Jul W3
(c) 1999 AAAS
File 610:BUSINESS WIRE 1999-2000/JAN 07
(c) 2000 BUSINESS WIRE.
File 612:JAPAN ECONOMIC NEWSWIRE(TM) 1984-2000/JAN 05
(c) 2000 KYODO NEWS
File 613:PR NEWSWIRE 1999-2000/JAN 07
(c) 2000 PR NEWSWIRE ASSOCIATION INC
File 621:GALE GROUP NEW PROD.ANNOU.(R) 1985-2000/JAN 07
(c) 2000 THE GALE GROUP
File 635:Business Dateline(R) 1985-1999/Nov 17
(c) 1999 Bell & Howell
File 484:Periodical Abstracts Plustext 1986-1999/Nov W3
(c) 1999 Bell & Howell
File 647:CMP COMPUTER FULLTEXT 1988-1999/DEC W3
(c) 1999 CMP
File 160:Gale Group PROMT(R) 1972-1989
(c) 1999 The Gale Group
File 810:Business Wire 1986-1999/Feb 28
(c) 1999 Business Wire
File 813:PR Newswire 1987-1999/Apr 30
(c) 1999 PR Newswire Association Inc
File 696:DIALOG TELECOM. NEWSLETTERS 1995-2000/JAN 07
(c) 2000 THE DIALOG CORP.

Set	Items	Description
S1	0	AU=(BOWMAN-AMUAH, M? OR BOWMAN-AMUAH M? OR BOWMAN AMUAH M? OR BOWMAN AMUAH, M?)
S2	9	BOWMAN()AMUAH
S3	5	RD (unique items)
S4	53	CO=VISUAL COMMUNICATIONS NETWORK?
S5	45	S4 NOT S2
S6	23	RD (unique items)

3/9/1 (Item 1 from file: 16)
DIALOG(R)File 16:GALE GROUP PROMT(R)

04248845 Supplier Number: 46222024 (THIS IS THE FULLTEXT)
High-Speed Wireless, Digital Internet Access 03/13/96
Newsbytes, pN/A
March 13, 1996
Language: English Record Type: Fulltext
Document Type: Newswire; General Trade
Word Count: 462
TEXT:
NEW` YORK, NEW YORK, U.S.A., 1996 MAR 13 (NB) -- Visual Communications Network is launching what it says is the world's first wireless multimedia, digital network capable of delivering full-motion video.

Operating at more than 180 times the speed of an integrated services digital network (ISDN) line, the company's high bandwidth network offers videoconferencing, data storage and retrieval systems, and Internet access to commercial customers.

"This is my dream, my vision," exclaimed Michel **Bowman -Amuah** , president of Visual Communications Network. "With this offering, we have eliminated the limitations of wireline fiber-optic requirements which have held electronic communications to a slow and boring presentation."

Bowman -Amuah explained to Newsbytes that logging in to his network required about $700 in hardware along with a $500 installation charge.

After the initial cost, users fees could vary depending on application, but the network could provide videoconferencing for about $0.95 per minute and unlimited Internet access for approximately $100 per month.

"Our wireless network can deliver data at 10 megabits-per-second (Mbps) and on the backbone operate at 1.6 gigabytes-per-second (GBps)," said **Bowman -Amuah** . While ISDN delivers an optimal 57.6 kilobits-per-second (Kbps) and commercial T-1 lines provide 1.54Mbps, Visual claims its wireless 10Mbps service allows users to experience full-motion video and Internet service which is television-like quality in smoothness and speed.

Addressing some of the difficulties with ISDN service, **Bowman -Amuah** said, "We offer a one-stop solution with full customer service. Ask people who have tried to connect an ISDN solution. They have to deal with the modem manufacturer, an ISDN provider and an Internet provider with ISDN capabilities. When you have a problem, you get bounced back and forth with three different companies trying to solve a single solution. We resolve all of those types of problems with a full service from installation to delivery of content." **Bowman -Amuah** , also said ISDN acceptance is slow because of the costs and installation problems. "The window of opportunity for ISDN is getting smaller as cable modems and wireless products and services are growing," he continued. "The average desktop user is unwilling to move from 28.8 kilobits-per-second modems to ISDN as long as the cost is not comparable. While they wait for lower cost ISDN, both cable and wireless are going to become more affordable and offer true high bandwidth."

"Our wireless service is still too expensive for the average desktop user, but it is affordable for commercial applications right now," said **Bowman -Amuah** .

Visual Communications' wireless network is the result of a strategic partnership with WorldCom (NASDAQ:WCOM) and alliances with Windstar Wireless (NASDAQ:WCII).

Visual president, Michel **Bowman -Amuah** , told Newsbytes that he plans to have wireless service from every major US city by the end of 1996.

(Patrick McKenna/19960312/Press Contact: Kathleen McDonnell, Visual Communications Network, 800-566-5665)

PUBLISHER NAME: Newsbytes News Network
COMPANY NAMES: *Visual Communications Network; Windstar Wireless; WorldCom
EVENT NAMES: *330 (Product information)
GEOGRAPHIC NAMES: *1USA (United States)

PRODUCT NAMES: *4811800 (Mo[redacted]e Communications Services)
INDUSTRY NAMES: BUSN (Any type of business); CMPT (Computers and Office Automation); TELC (Telecommunications)
NAICS CODES: 513322 (Cellular and Other Wireless Telecommunications)
SPECIAL FEATURES: COMPANY

3/9/2 (Item 2 from file: 16)
DIALOG(R)File 16:GALE GROUP PROMT(R)

04245969 Supplier Number: 46217289 (THIS IS THE FULLTEXT)
WIRELESS INTERNET ACCESS!!
PR Newswire, p0311NYM112
March 11, 1996
Language: English Record Type: Fulltext
Document Type: Newswire; Trade
Word Count: 333
TEXT:
NEW YORK, March 11 /PRNewswire/ -- Visual Communications Network, Ltd. today launched what may be the nation's first wireless multi-media data service network. This service allows users to interact with applications such as television-like, two-way video conferencing and electronic mail and data storage/retrieval systems from locations over a wireless network. When asked to comment on the implications of this announcement, the President, Michel **Bowman -Amuah** stated that "We have extended the reach of personal communications and eliminated the limitation of wireline fiber- optic requirements for efficient delivery of our high bandwidth products and services."

In an unprecedented strategic partnership relationship with LDDS-WorldCom (Nasdaq-NNM: WCOM) and alliances with Windstar Wireless (WCII) (Nasdaq-NNM: WCII), TCG, ICG and NMS, Visual Communications Network expects to have network coverage in every major Metropolitan U.S. city by the end of 1996. "Our strategic alliances provide us with the network capacity, redundancy and reliability to which users of voice services have become accustomed, but have never been given in a high bandwidth on demand basis," said **Bowman -Amuah** . When asked about the availability and cost of these services, **Bowman -Amuah** replied "Prices depend on location and vary from application to application however a good rule of thumb is $350 monthly for 30 Frames per second Video conferencing and high speed internet access from any one of 135 major metropolitan U.S. cities."

Visual Communications Network is a full service multi-media communications provider offering videoconferencing at $0.95 cents per minute and multimedia internet products to business and residential users. VCN is a subsidiary of International Investment Group, Ltd. (IIGR NASD Bulletin Board symbol).

For information regarding the relationships between the companies, contact Jim Royal, Vice President of WorldCom and Pete Racelis, Senior Vice President of Sales and Marketing of Windstar Wireless.

-0- 3/11/96

/CONTACT: Kathleen McDonnell of Visual Communications Network, 800-566-LOOK or 800-566-5665/

(WCOM WCII)

CO: Visual Communications Network, Ltd.; Windstar Wireless
ST: New York
IN: CPR TLS
SU: PDT

NT-SR
-- NYM112 --
0987 03/11/96 17:21 EST

PUBLISHER NAME: PR Newswire Association, Inc.
COMPANY NAMES: *Visual Communications Network

EVENT NAMES: *360 (Services [illegible]ormation)
GEOGRAPHIC NAMES: *1USA (United States)
PRODUCT NAMES: *4811500 (Specialized Telecommunication Services)
INDUSTRY NAMES: BUS (Business, General); BUSN (Any type of business)
NAICS CODES: 51331 (Wired Telecommunications Carriers)
SPECIAL FEATURES: COMPANY

3/9/3 (Item 1 from file: 15)

01220188 98-69583
Editorial
Hoke, Henry R; Hoke, Henry Reed III
Direct Marketing v59n1 PP: 80 May 1996 CODEN: DIMADI ISSN: 0012-3188
JRNL CODE: DIM
DOC TYPE: Journal article LANGUAGE: English LENGTH: 1 Pages
WORD COUNT: 1039

ABSTRACT: The Greater Raleigh Chamber of Commerce has launched a local Intranet site for use by businesses located within the famous Research Triangle of Chapel Hill, Durham and Raleigh, North Carolina. Meanwhile, Visual Communications Network's new system enables videoconferencing - and more - right on personal computers.

TEXT: Sidebar: THE INTERNET HOLDS GREAT FASCINATION for this oldtimer. I can hear you say: "What's new?" Says I: "The velocity of this innovation, that's what!" It's breathtaking. Following a recent luncheon meeting of The Carolinas Direct Marketing Association at Research Triangle Park's Holiday Inn, Demi and I trundled down to Raleigh's Civic Center to tour some 300 exhibits at the one-day Small Business Expo '96. The Center was buzzing with business executives, anxious to catch up on the new toys being displayed. Bill Gates set the tone in an early morning address in the auditorium. He waxed enthusiastic about the Internet and the Intranet, the internal networks within a business connecting employees with each other anywhere on earth.

The Greater Raleigh Chamber of Commerce has developed an extension of this idea for businesses located within the famous Research Triangle of Chapel Hill, Durham and Raleigh, North Carolina. The Chamber has launched a local Intranet site for use by Triangle businesses-thousands of them-on the theory that it's plain dumb to buy business supplies and services around the country when most of them can be had locally from neighboring businesses. The Chamber calls its idea Buy Local Online Service. The site's administrator provided me a diskette for my Mac 540c, so that I could surf the Triangle from my easychair in Tryon.

Businesses which sign on provide Buy Local with their SIC code. This allows sorting/indexing by kind of business. Fellow businessmen can search the database by SIC and/or alphabetically, by name of firm. Having entered search criteria, the system provides a list of business names which qualify. Surfers click and access individual sites, read about products and services offered. Surfers click to reveal whether they're buying or selling. If buying, they can read what's available. If selling, the Intranet salesperson can read an overview of business, visualize their needs, then craft an e-mail and send. Or pick up the phone and reach the purchasing influence. Cost? Participants pay $100 a year or less depending on membership and have e-mail throughout the Triangle. Talk to Nita Fulbright, vice president/membership (919/664 7031).

Raleigh's newspaper, The News & Observer, will not be left behind. They've just announced a new online service for classified ads. Eventually the paper will provide hyperlinks to advertisers' own pages. You can ask questions of Chris Brewer, classified new media developer online at cbrewer@nando.com or by phone 919/836-5796. The Chamber might well offer members Martin Baier's new book How To Find and Cultivate Customers Through Direct Marketing, with a forward by Bob Stone and published by NTC. Martin

deals with the very essence of direct marketing by providing a blueprint for building and maintaining any company's customer file or marketing database. It's a solid book. You should have it. Cordially,

MY LONG-TIME FRIEND, fraternity brother and the original manager of Ideas In Sound, Bert Cunningham IN(president, Howard Blankman, Inc., Jericho, NY), wrote me a kind note last month, asking me to attend a special meeting on visual communications at the Garden City Hotel.
Bert, you see, asks me to meet him from time to time, not only to re-live old times, but also to see where we each are headed and what new marketing ideas we've both seen and heard about. Both of us being schooled in communications and marketing and having the same interests in life has lent itself to a great friendship and great conversation. Anyway, the invitation comes in, I respond, by phone to the accept the invitation, write it down on my desk calendar and I'm ready.

It was some bash. Hundreds of people (new people!), hors d'oeuvres and drinks, with lots of mingling. As the demonstration began, I couldn't help but be reminded of a presentation I'd heard in 1985 at the Montreux Symposium. There I listened to the vice president of marketing for Sony Corporation of Europe speak about the digital world that was about to explode. She said, "Your world, as marketers, will change when you can merge data, text, audio and video all at the same time, from any location to another."

Back to the meeting. Imagination became reality as Long Island's business and government leaders viewed a new multimedia communications breakthrough enabling videoconferencing-and more-right on personal computers.

This system is the introduction of a local and wide area networks, using personal computers for videoconferencing. The system also allowing users to view, edit and share online documentation, as well as access to the Internet, e-mail, color fax-mail and transfer media. The key issue here is that this system runs at 30 frames per second, which is real-time video with synchronized audio. Now, live and taped presentations, text and data can be easily transmitted via your computer at real time. Until now, the problems with computers sending audio and video signals was that the signals tended to lag and break up. As InterMediaNet states, not anymore. Michael **Bowman - Amuah** , president and inventor, Visual Communications Network, stated that this new revolutionary system eliminates those old-time problems related to video conferencing, and costs only 95 cents per minute.

"Wait a minute here! You gotta be kidding!" I thought to myself as I was watching live feeds from Totowa, New Jersey, Colorado Springs, Colorado and Garden City, New York all at once. "There has to be a catch here."

Yes, there were some technical problems. That comes with the territory when launching a new system. Yes, you obviously have to have a T- lines in at those locations, programs and a camera or two. But the system has arrived. It's taken eleven years since that speech in Montreux to become reality. The electronic communications world is here to stay.
Like it or not, instant communications is where we all are going. It is exciting to see technology still leading the way for all types of businesses to use. It is what keeps us all looking forward. If your interested and need more information, contact Doug Winkler, vice president of marketing, Visual Communications Network Ltd., 100 Ring Road West, Suite 204, Garden City, NY 11530-516/741-1900 (Fax 516/741-3819).

Spring is here. I feel better-and more confident-that the technology I have been chasing for all this time is finally here and will be part of the business I'm in.

GEOGRAPHIC NAMES: US

DESCRIPTORS: Editorials; Direct marketing; Technological change; Intranets; Video teleconferencing

CLASSIFICATION CODES: 9190 (CN=United States); 7300 (CN=Sales & Billing);
5250 (CN=Telecommunications systems)

3/9/4 (Item 1 from file: 9)
DIALOG(R)File 9:BUSINESS & INDUSTRY(R)

01435339 (THIS IS THE FULLTEXT)
High-Speed Wireless, Digital Internet Access
(Visual Communications Network's wireless, high bandwidth digital network operates at over 180 times the speed of an ISDN line)
Newsbytes News Network, p N/A
March 13, 1996
DOCUMENT TYPE: Journal (United States)
LANGUAGE: English RECORD TYPE: Fulltext
WORD COUNT: 452

TEXT:
NEW' YORK, NEW YORK, U.S.A., 1996 MAR 13 (NB) -- Visual Communications Network is launching what it says is the world's first wireless multimedia, digital network capable of delivering full-motion video. Operating at more than 180 times the speed of an integrated services digital network (ISDN) line, the company's high bandwidth network offers videoconferencing, data storage and retrieval systems, and Internet access to commercial customers.

"This is my dream, my vision," exclaimed Michel **Bowman -Amuah** , president of Visual Communications Network. "With this offering, we have eliminated the limitations of wireline fiber-optic requirements which have held electronic communications to a slow and boring presentation."

Bowman -Amuah explained to Newsbytes that logging in to his network required about $700 in hardware along with a $500 installation charge. After the initial cost, users fees could vary depending on application, but the network could provide videoconferencing for about $0.95 per minute and unlimited Internet access for approximately $100 per month.

"Our wireless network can deliver data at 10 megabits-per-second (Mbps) and on the backbone operate at 1.6 gigabytes-per-second (GBps)," said **Bowman -Amuah** . While ISDN delivers an optimal 57.6 kilobits-per-second (Kbps) and commercial T-1 lines provide 1.54Mbps,

Visual claims its wireless 10Mbps service allows users to experience full-motion video and Internet service which is television-like quality in smoothness and speed.

Addressing some of the difficulties with ISDN service, **Bowman -Amuah** said, "We offer a one-stop solution with full customer service. Ask people who have tried to connect an ISDN solution. They have to deal with the modem manufacturer, an ISDN provider and an Internet provider with ISDN capabilities. When you have a problem, you get bounced back and forth with three different companies trying to solve a single solution. We resolve all of those types of problems with a full service from installation to delivery of content."

Bowman -Amuah , also said ISDN acceptance is slow because of the costs and installation problems. "The window of opportunity for ISDN is getting smaller as cable modems and wireless products and services are growing," he continued. "The average desktop user is unwilling to move from 28.8 kilobits-per-second modems to ISDN as long as the cost is not comparable. While they wait for lower cost ISDN, both cable and wireless are going to become more affordable and offer true high bandwidth."

"Our wireless service is still too expensive for the average desktop user, but it is affordable for commercial applications right now," said **Bowman -Amuah** .

Visual Communications' wireless network is the result of a strategic

partnership with WorldCom (NAS[redacted]:WCOM) and alliances with Wind[redacted]r Wireless (NASDAQ:WCII).

Visual president, Michel **Bowman -Amuah** , told Newsbytes that he plans to have wireless service from every major US city by the end of 1996. (Patrick McKenna/19960312/Press Contact: Kathleen McDonnell, Visual Communications Network, 800-566-5665)

COMPANY NAMES: VISUAL COMMUNICATIONS NETWORK
INDUSTRY NAMES: Mobile communications; Telecom services; Telecommunications
PRODUCT NAMES: Specialized mobile radio (SMR) services (481294)
CONCEPT TERMS: All product and service information; Product introduction
GEOGRAPHIC NAMES: North America (NOAX); United States (USA)

3/9/5 (Item 1 from file: 813)
DIALOG(R)File 813:PR Newswire

1000593 NYF075

United States Government Joint Chiefs Of Staff Select VCN As Exclusive Desktop Video And Document Conferencing Network Provider.

DATE: September 27, 1996 15:13 EDT WORD COUNT: 371

NEW YORK Sept. 27 Visual Communications Network Ltd. a subsidiary of InterMedia Net Inc. (Nasdaq-BB: BNET) today announced a number of significant events that a Company spokesman referred to as the beginning of our full frontal attack on the Multimedia, Internet and Video Communications market. The Company has successfully completed its ATM trials in San Francisco, Dallas, New York and Colorado and have signed up users and resellers in Colorado Springs, San Francisco and Dallas. Some of the highlights include, Forefront Technologies' internet subsidiary, High Plains Internet, HP.NET: out of Colorado Springs, Colorado signed a $45,000 agreement; in another deal the San Francisco office of VCN signed up GRQ International Consulting Inc. and Red Box Advertising on multi year web hosting and access service agreements totaling approximately $75,000. Similarly VCN's Video Products Division announced hardware and software systems sales of approximately $200,000 with First Choice Marketing, Ardee Eyewear, Message Com Plus and Nendor Engineering. The most high profile account we have landed to date is the Joint Logistics Services Committee who represent the Joint Chiefs of Staff and other government and military agencies said VCN President, Michel **Bowman -Amuah** . The Joint chiefs of staff selected VCN as the exclusive desktop video and document conferencing network provider. In a deal valued at $120,000 with the potential of $2.5M in equipment revenue and additional related recurring network revenue of $6,000 per seat annually. Finally, Dallas, Texas based systems integrator CompuCom (NASDAQ: CMPC), the nation's fourth largest systems integrator, agrees to a 60 day 9 site ATM video network trial over InterMedia-Net, which when successfully completed would convert to an order of $85,000 in video gateway servers and ATM based multi-point controllers. CompuCom anticipates the use of the ATM network to link all their regional hub location from California to Massachusetts for video collaboration, data communications and high speed Internet access.

Visual Communications Network Ltd., a subsidiary of InterMedia Net Inc. (NASD bulletin board Symbol BNET) develops and integrates multimedia video and
data systems over their ATM network backbone InterMedia.Net and can be contacted at 1-800-566-5665 e-mail http://www.5661ook.com

SOURCE Visual Communications Network Ltd.

CONTACT: Michel **Bowman Amuah** of Visual Communications Network Ltd., 800-566-5665

(BNET)

COMPANY NAME: VISUAL COMMUNICATIONS NETWORK LTD.
TICKER SYMBOL: CMPC (NDQ); BNET (NDQ)
PRODUCT: INTERNET, MULTIMEDIA, ONLINE (MLM)
STATE: NEW YORK (NY)
SECTION HEADING: BUSINESS
?

6/3,K/1 (Item 1 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01254660 SUPPLIER NUMBER: 06927365 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Making moving pictures. (desktop animation packages)
Einstein, Jeff
PC Magazine, v7, n16, p241(6)
Sept 27, 1988
ISSN: 0888-8507 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 1530 LINE COUNT: 00126

...COMPANY NAMES: **Visual Communications Network Inc**

6/3,K/2 (Item 2 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01247520 SUPPLIER NUMBER: 06988523 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Desktop-presentation tools fit range of needs.
Picarille, Lisa
PC Week, v5, n38, p94(1)
Sept 19, 1988
ISSN: 0740-1604 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 716 LINE COUNT: 00058

...COMPANY NAMES: **Visual Communications Network Inc**

6/3,K/3 (Item 3 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01220714 SUPPLIER NUMBER: 06151852
Graphics cornucopia: VCN Concorde graphics program. (Software Review) (evaluation)
Jantz, Richard
Publish, v3, n1, p64(3)
Jan, 1988
DOCUMENT TYPE: evaluation ISSN: 0897-6007 LANGUAGE: ENGLISH
RECORD TYPE: ABSTRACT

COMPANY NAMES: **Visual Communications Network Inc ...**

6/3,K/4 (Item 4 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01207485 SUPPLIER NUMBER: 04749528 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Concorde. (Software Review) (animating business graphics) (evaluation)
Rosch, Winn L.
PC Magazine, v6, n5, p266(3)
March 10, 1987
DOCUMENT TYPE: evaluation ISSN: 0888-8507 LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 1197 LINE COUNT: 00095

COMPANY NAMES: **Visual Communications Network Inc ...**

6/3,K/5 (Item 5 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01207484 SUPPLIER NUMBER: 04749527 (USE FORMAT 7 OR 9 FOR FULL TEXT)

ExecuVision. (Software Review) (Animating business graphics) (evaluation)
Rosch, Winn L.
PC Magazine, v6, n5, p263(2)
March 10, 1987
DOCUMENT TYPE: evaluation ISSN: 0888-8507 LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 473 LINE COUNT: 00038

COMPANY NAMES: **Visual Communications Network Inc ...**

6/3,K/6 (Item 6 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01195338 SUPPLIER NUMBER: 04683862
New graphics upgrade offers 2 major modules. (Concorde 2.0) (product announcement)
PC Week, v4, n9, p16(1)
March 3, 1987
DOCUMENT TYPE: product announcement ISSN: 0740-1604 LANGUAGE: ENGLISH RECORD TYPE: ABSTRACT

COMPANY NAMES: **Visual Communications Network Inc ...**

6/3,K/7 (Item 7 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01195195 SUPPLIER NUMBER: 06149070
Graphics packages for the PC and compatibles - two of the best. (Software Review) (evaluation)
Hannum, David L.
IEEE Micro, v7, n1, p78(2)
Feb, 1987
DOCUMENT TYPE: evaluation ISSN: 0272-1732 LANGUAGE: ENGLISH
RECORD TYPE: ABSTRACT

COMPANY NAMES: **Visual Communications Network Inc ...**

6/3,K/8 (Item 8 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01192556 SUPPLIER NUMBER: 04733391
Slide show software. (Software Review) (evaluation)
Milburn, Ken
InfoWorld, v9, n12, p37(3)
March 23, 1987
DOCUMENT TYPE: evaluation ISSN: 0199-6649 LANGUAGE: ENGLISH
RECORD TYPE: ABSTRACT

...COMPANY NAMES: **Visual Communications Network Inc**

6/3,K/9 (Item 9 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01183894 SUPPLIER NUMBER: 05154522
VCN Concorde. (Software Review) (Visual Communications Network Inc.'s VCN Concorde computer graphics software) (evaluation)
Mau, Ernest E.
Online Today, v6, n2, p42(2)
Feb, 1987
DOCUMENT TYPE: evaluation ISSN: 0891-4672 LANGUAGE: ENGLISH

RECORD TYPE: ABSTRACT

COMPANY NAMES: **Visual Communications Network Inc ...**

6/3,K/10 (Item 10 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01178347 SUPPLIER NUMBER: 04366776 (USE FORMAT 7 OR 9 FOR FULL TEXT)
3 graphics programs animate screen images.
Rosch, Winn L.
PC Week, v3, n39, p131(2)
Sept 30, 1986
ISSN: 0740-1604 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 981 LINE COUNT: 00080

...COMPANY NAMES: **Visual Communications Network Inc**

6/3,K/11 (Item 11 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01177676 SUPPLIER NUMBER: 04430880 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Concorde package stands out. (Software Review) (evaluation)
Rosch, Winn
PC Week, v3, n42, p83(4)
Oct 21, 1986
DOCUMENT TYPE: evaluation ISSN: 0740-1604 LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 1745 LINE COUNT: 00138

COMPANY NAMES: **Visual Communications Network Inc ...**

6/3,K/12 (Item 12 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01169777 SUPPLIER NUMBER: 04537622
Show-stopping slides. (Software Review) (IBM PC-based slide presentations) (evaluation)
Jantz, Richard
PC World, v4, n11, p260(5)
Nov, 1986
DOCUMENT TYPE: evaluation ISSN: 0737-8939 LANGUAGE: ENGLISH
RECORD TYPE: ABSTRACT

COMPANY NAMES: **Visual Communications Network Inc ...**

6/3,K/13 (Item 13 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01169412 SUPPLIER NUMBER: 04586608
VCN Concorde. (Software Review) (graphics software) (evaluation)
Rothfeder, Jeffrey
PC Magazine, v5, n20, p250(3)
Nov 25, 1986
DOCUMENT TYPE: evaluation ISSN: 0888-8507 LANGUAGE: ENGLISH
RECORD TYPE: ABSTRACT

COMPANY NAMES: **Visual Communications Network Inc ...**

6/3,K/14 (Item 14 from file: 275)

DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01169200 SUPPLIER NUMBER: 04421328
Mapmaking with graphics packages. (Software Review) (evaluation)
Day, Carol Olsen
PC Magazine, v5, n16, p220(2)
Sept 30, 1986
DOCUMENT TYPE: evaluation ISSN: 0888-8507 LANGUAGE: ENGLISH
RECORD TYPE: ABSTRACT

...COMPANY NAMES: **Visual Communications Network Inc**

6/3,K/15 (Item 15 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01166528 SUPPLIER NUMBER: 04321782
Business graphics claims to do everything. (Software Review) (evaluation)
Robertson, Barbara
InfoWorld, v8, n33, p42(2)
Aug 18, 1986
DOCUMENT TYPE: evaluation ISSN: 0199-6649 LANGUAGE: ENGLISH
RECORD TYPE: ABSTRACT

COMPANY NAMES: **Visual Communications Network Inc ...**

6/3,K/16 (Item 16 from file: 275)
DIALOG(R)File 275:GALE GROUP COMPUTER DB(TM)
(c) 2000 THE GALE GROUP. All rts. reserv.

01044258 SUPPLIER NUMBER: 00561048
Visual Comm. Inks IBM Pact.
Computer Retail News, n6, p73
July 2, 1984
ISSN: 0744-673X LANGUAGE: ENGLISH RECORD TYPE: ABSTRACT

COMPANY NAMES: **Visual Communications Network ...**

6/3,K/17 (Item 1 from file: 16)
DIALOG(R)File 16:GALE GROUP PROMT(R)
(c) 2000 THE GALE GROUP. All rts. reserv.

04390649 Supplier Number: 46440712 (USE FORMAT 7 FOR FULLTEXT)
INTERMEDIA NET, INC. ANNOUNCES ITS FIRST MULTI-ACCESS, MULTI-MEDIA DATA SERVICE NETWORK
PR Newswire, p0603NYM049
June 3, 1996
Language: English Record Type: Fulltext
Document Type: Newswire; Trade
Word Count: 463

COMPANY NAMES: Intermedia Net; **Visual Communications Network**

6/3,K/18 (Item 1 from file: 15)
DIALOG(R)File 15:ABI/INFORM(R)
(c) 1999 Bell & Howell. All rts. reserv.

00422776 88-39609
Tooning In on 'Desktop Video'
James, Watson Saint
Advertising Age v59n43 PP: 48, 66 Oct 10, 1988
ISSN: 0001-8899 JRNL CODE: ADA

6/3,K/19 (Item 2 from file: 15)
DIALOG(R)File 15:ABI/INFORM(R)

00331463 86-31877
Product Reviews: All-in-One Graphics
O'Malley, Christopher
Personal Computing v10n9 PP: 144, 146 Sep 1986
ISSN: 0192-5490 JRNL CODE: PSC

6/3,K/20 (Item 1 from file: 148)
DIALOG(R)File 148:GALE GROUP TRADE & INDUSTRY DB

08158959 SUPPLIER NUMBER: 17486591 (USE FORMAT 7 OR 9 FOR FULL TEXT)
VISUAL COMMUNICATIONS NETWORK ANNOUNCES FAVORABLE RESPONSE PERTAINING TO RECENT PATENT APPLICATION
PR Newswire, p927NY095
Sep 27, 1995
LANGUAGE: English RECORD TYPE: Fulltext
WORD COUNT: 266 LINE COUNT: 00030

COMPANY NAMES: **Visual Communications Network Inc ...**

6/3,K/21 (Item 2 from file: 148)
DIALOG(R)File 148:GALE GROUP TRADE & INDUSTRY DB

03425060 SUPPLIER NUMBER: 06301942
Desktop video: new toy or effective communication tool?
Gordon, Gloria
Communication World, v5, n4, p20(2)
March, 1988
ISSN: 0744-7612 LANGUAGE: ENGLISH RECORD TYPE: ABSTRACT

COMPANY NAMES: **Visual Communications Network Inc ...**

6/3,K/22 (Item 1 from file: 621)
DIALOG(R)File 621:GALE GROUP NEW PROD.ANNOU.(R)

01033444 Supplier Number: 39969416 (USE FORMAT 7 FOR FULLTEXT)
VCN RELEASES CONCORDE VERSION 2.0 WITH MAJOR ENHANCEMENTS THAT MAKE IT UNIQUE IN THE GRAPHICS SOFTWARE MARKETPLACE
PR Newswire, pN/A
Feb 16, 1987
Language: English Record Type: Fulltext
Document Type: Newswire; Trade
Word Count: 1050

COMPANY NAMES: **Visual Communications Network**

6/3,K/23 (Item 1 from file: 484)
DIALOG(R)File 484:Periodical Abstracts Plustext

00258482
Tooning in on 'Desktop Video'
Saint James, Watson
Advertising Age (ADA), v59 n43, p48, 66
Oct 10, 1988
ISSN: 0001-8899 JOURNAL CODE: ADA
DOCUMENT TYPE: Product Review-Comparative

LANGUAGE: English RECORD TYPE: Abstract
LENGTH: Medium (10-30 col inches)

COMPANY NAMES: **Visual Communications Network Inc ...**

WEST

Generate Collection

L1: Entry 1 of 2 File: USPT Feb 24, 1998

US-PAT-NO: 5721908
DOCUMENT-IDENTIFIER: US 5721908 A

TITLE: Computer network for WWW server data access over internet

DATE-ISSUED: February 24, 1998

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Lagarde; Konrad Charles	Milford	CT	N/A	N/A
Rogers; Richard Michael	Beacon	NY	N/A	N/A

ASSIGNEE INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY	N/A	N/A	02

APPL-NO: 8/ 474571
DATE FILED: June 7, 1995

INT-CL: [6] G06F 17/30, G06F 17/40, G06F 17/28

US-CL-ISSUED: 395/610; 395/601, 395/602, 395/604, 395/329, 395/335, 395/200.09
US-CL-CURRENT: 707/10; 345/329, 345/335, 707/1, 707/2, 707/4, 709/202

FIELD-OF-SEARCH: 395/600, 395/610, 395/601, 395/602, 395/604, 395/329, 395/335, 395/200.09

REF-CITED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4274139	June 1981	Hodgkinson	364/200
☐	4468728	August 1984	Wang	364/200
☐	4604710	August 1986	Amezcua et al.	364/900
☐	4714989	December 1987	Billings	364/200
☐	4714995	December 1987	Materna et al.	364/200
☐	4774655	September 1988	Kollin et al.	364/200
☐	5093911	March 1992	Parks et al.	395/615
☐	5179652	January 1993	Rozmanith et al.	395/155
☐	5181017	January 1993	Frey, Jr. et al.	340/825
☐	5224098	June 1993	Bird et al.	370/94.1
☐	5241625	August 1993	Epard et al.	385/502
☐	5278978	January 1994	Demers et al.	395/600
☐	5297249	March 1994	Bernstein	395/356
☐	5307456	April 1994	Mackay	395/328
☐	5355472	October 1994	Lewis	395/607
☐	5530852	June 1996	Meske et al.	395/600

OTHER PUBLICATIONS

Christopher G. Thomas "A framework for intergrating agents in the World Wide Web" GMD FIT, HCI Research Division pp. 84-86 May 1995.
James Powell "Creating a Hypertext Library Information System" Virginia Polytechnic Institute and State University pp. 59-66 Feb. 1994.
Marcus et al. "User-Interface Developments for the Nineties" IEEE pp. 49-57 Sep. 1991.
Armbruster et al. "Broadband Multimedia Applications Using ATM Networks" IEEE pp. 1382-1396 vol. 10, No. 9 Dec. 1992.
"Harvest: A Scalable, Customizable Discovery and Access System" by Bowman et al., Aug. 26, 1994, Technical Report CU-CS-732-94.
HTML and MOSAIC: A taste for more, "Use of Hyper Text Mark-up Language (HTML) for the development of interactive multimedia through World Wide Web (WWW)" by P. L. Linde, INET 94, pp. 212-1 -212-7.
"Developing Applications with OpenDIS Access Service", Metaphor Data Interpretation System Release 2.0, Metaphor Inc., 1st Edition, Sep.1994.

ART-UNIT: 237

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Corrielus; Jean M.

ATTY-AGENT-FIRM: Augspurger; Lynn L.

ABSTRACT:

A World Wide Web browser makes requests to web servers on a network which receive and fulfill requests as an agent of the browser client, organizing distributed sub-agents as distributed integration solution (DIS) servers on an intranet network supporting the web server which also has an access agent servers accessible over the Internet. DIS servers execute selected capsule objects which perform programmable functions upon a received command from a web server control program agent for retrieving, from a database gateway coupled to a plurality of database resources upon a single request made from a Hypertext

document, requested information from multiple data bases located at different types of databases geograhically dispersed, performing calculations, formatting, and other services prior to reporting to the web browser or to other locations, in a selected format, as in a display, fax, printer, and to customer installations or to TV video subscribers, with account tracking.

26 Claims, 11 Drawing figures
Exemplary Claim Number: 23
Number of Drawing Sheets: 9

BRIEF SUMMARY:

COPYRIGHT AUTHORIZATION

FIELD OF THE INVENTION

This invention is related to computers and computer systems and particularly to a method and system for use of the World Wide Web and other sources of information and for utilization of existing equipment advantageously for web server data access over networks and the Internet.

RELATED APPLICATIONS

This application entitled "Computer Network for WWW Server Data Access over Internet", is related to other United States of America Patent applications filed concurrently herewith, and specifically to the applications entitled "A Service Agent for Fulfilling requests of a Web Browser", U.S. Ser. No. 08/474,576, filed Jun. 7, 1995; and "A Sub-Agent Service Agent for Fulfilling Requests of a Web Browser", U.S. Ser. No. 08/474,575, filed Jun. 7, 1995; and "A Web Browser System", U.S. Ser. No. 08/474,481, filed Jun. 7, 1995; and "A Method for Fulfilling Requests of a Web Browser" U.S. Ser. No. 08/479,577, filed Jun. 7, 1995; and "A Method for Distributed Task Fulfillment of Web Browser Requests", U.S. Ser. No. 08/474,572, filed Jun. 7, 1995.

These applications have a common assignee, International Business Machines Corporation, Armonk, N.Y.

GLOSSARY OF TERMS

While dictionary meanings are also implied by certain terms used here, the following glossary of some terms may be useful.

World Wide Web (WWW) The Internet's application that lets people seeking information on the Internet switch from server to server and database to database by clinking on highlighted words or phrases of interest. An Internet WWW server supports clients and provides information.

Home page A multi-media table of contents that guides a web user to stored information about an organization on the Internet.

Gopher A menu-based search scheme, which as developed at the University of Minnesota, lets a user reach a destination on the Internet by selecting items from a series of text menus.

Access Agent A logical component that provides support for different access protocols and data streams--Frame Relay, HDLC (High Data Link Control) CBO (Continuous bit Operations, ATM (Asychronous Transfer Mode), or TCP/IP.

Application Processing Agent A data processing agent running in a server data processing system which performs tasks based on received requests from a client in a distributed environment. In our preferred embodiment, our application processing agent for database retrieval is our DIS server, a data

interpretation system server and database gateway which is coupled to our web server HTTPD via a network. In our preferred embodiment an application processing agent employs executable object programs as command file objects, which in the preferred embodiment are capsule objects.

Client A client is a computer serviced by the server which provides commands to the server.

Data Interpretation System (DIS). IBM's object oriented decision support tool.

Capsule A DIS capsule is a program created by a DIS programmer and executed in the DIS environment. A DIS capsule is a preferred example of a capsule object. A capsule object is a specialized form of a command file (which is a list of commands to be executed, as in an EXEC or *.BAT batch file. The capsule object is created with an object environment, as is supplied by IBM's DIS. Other object environments are IBM's SOM and DSOM, and Microsoft's COM environment.

Internet The connection system that links computers worldwide in a web.

Server A machine which supports one or more clients and is part of the web. Any computer that performs a task at the command of another computer is a server.

Slip or PPP connection. Serial-line Internet protocol and point-to-point protocol, respectively, for providing a full access connection for a computer to the Internet.

TCP/IP Transmission control protocol/Internet protocol. A packet switching scheme the Internet uses to chop, route, and reconstruct the data it handles, from e-mail to video.

InterNetwork Routing (INR) The link between systems which routes data from one physical unit to another according to the applicable protocol. The protocol will employ a URL address for Internet locations.

URL Universal resource locater, a Web document version of an e-mail address. URLs are very cumbersome if they belong to documents buried deep within others. They can be accessed with a Hyperlink.

Web browser An program running on a computer that acts as an Internet tour guide, complete with pictorial desktops, directories and search tools used when a user "surfs" the Internet. In this application the Web browser is a client service which communicates with the World Wide Web.

HTTPD An IBM OS/2 Web Server or other server having Hypertext Markup Language and Common Gateway Interface. In our preferred embodiment, the HTTPD incorporates our control program agent and is supported by an access agent which provides the hardware connections to machines on the intranet and access to the Internet, such as TCP/IP couplings.

HTTP Hypertext transfer protocol Hypertext transfer protocol. At the beginning of a URL "http:" indicates the file contains hyperlinks.

Hyperlink A network address embedded in a word, phrase, icon or picture that is activated when you select the highlighted tidbit. Information about that item is currently retrieved to the client supporting a Web browser.

HyperText Markup Language (HTML) HTML is the language used by Web servers to create and connect documents that are viewed by Web clients. HTML uses Hypertext documents. Other uses of Hypertext documents are described in U.S. Pat. Nos. 5,204,947, granted Apr. 20, 1993 to Bernstein et al.; 5,297,249, granted Mar. 22, 1994 to Bernstein et al.; 5,355,472, granted Oct. 11, 1994 to Lewis; all of which are assigned to International Business Machines Corporation, and which are referenced herein.

BACKGROUND OF THE INVENTION

The Internet is not a single network, it has no owner or controller, but is an unruly network of networks, a confederation of many different nets, public and private, big and small, that have agreed to connect to one another. An intranet

is a network which is restricted and while it may follow the Internet protocol, none or only part of the network available from outside a "firewall" surrounding the intranet is part of the agreed connection to the Internet. The composite network represented by these networks relies on no single transmission medium, bi-directional communication can occur via satellite links, fiber-optic trunk lines, phone lines, cable TV wires and local radio links. When your client computer logs onto the Internet at a university, a corporate office or from home, everything looks local, but the access to the network does cost time and line charges.

Until recently, "cruising or surfing" the Internet was a disorienting, even infuriating experience, something like trying to navigate without charts. The World Wide Web, a sub-network of the Internet, introduced about two years ago, made it easier by letting people jump from one server to another simply by selecting a highlighted word, picture or icon (a program object representation) about which they want more information--a maneuver known as a "hyperlink". In order to explore the WWW today, the user loads a special navigation program, called a "Web browsed" onto his computer. While there are several versions of Web browsers, IBM's example is the new WebExplorer which offers users of IBM's OS/2 Warp system software a consistent, easy to use desktop of pictorial icons and pull down menus. As part of a group of integrated applications available from IBM for OS/2 Warp called the IBM Internet Connection, lets users log onto the Internet.

To this point the World Wide Web (Web) provided by Internet has been used in industry predominately as a means of communication, advertisement, and placement of orders. As background for our invention there now exists a number of Internet browsers. Common examples are NetScape, Mosaic and IBM's Web Explorer. Browsers allow a user of a client to access servers located throughout the world for information which is stored therein and provided to the client by the server by sending files or data packs to the requesting client from the server's resources. An example of such a request might be something called GSQL (get SQL) which was a NCSA language and CGI server program developed to getting textual results for a client caller. Developed by Jason Ng at the University of Illinois, this document provided a way to map SQL forms against a database, and return the textual results to the client caller. This system is unlike the present invention, and presents difficulties which are overcome by our described system.

These servers act as a kind of Application Processing Agent, or (as they may be referred to) an "intelligent agent", by receiving a function request from a client in response to which the server which performs tasks, the function, based on received requests from a client in a distributed environment. This function shipping concept in a distributed environment was first illustrated by CICS as a result of the invention described in U.S. Pat. No. 4,274,139 to Hodgkinson et al. This kind of function, illustrated by CICS and its improvements, has been widely used in what is now known as transaction processing. However, servers today, while performing many functions, do not permit the functions which we have developed to be performed as we will describe.

Now, "surfing" the Internet with the WWW is still a time consuming affair, and the information received is not generally useful in the form presented. Even with 14,400 baud connection to the Internet much line time is tied up in just keeping going an access to the Internet, and the users don't generally know where to go. Furthermore the coupling of resources available on a company's intranet and those available on the Internet has not been resolved. There is a need to reduce gateways, make better use of existing equipment, and allow greater and more effective usage of information which is resident in many different databases on many different servers, not only within a homogeneous network but also via the Internet and heterogeneous network systems.

The problems with creating access to the world via the Internet and still to allow internal access to databases has been enormous. However, the need for a system which can be used across machines and operating systems and differing gateways is strongly felt by users of the Internet today. Anyone who has spent hours at a WWW browser doing simple task knows how difficult it still is to navigate thorough arcane rules without knowing where to go and even if you know what you are doing spending hours doing routine tasks. Many needs exist. As one

important instance, until now we know of no way to access data on multiple databases of different types using a single user request from a client. This and other difficulties are solved by our invention.

SUMMARY OF THE INVENTION

In accordance with our invention needless user intervention is eliminate or greatly reduced with a Web server supports an HTTPD which is provided with the capabilities of our control program agent which organizes sub-agents supporting command file objects or capsules to perform tasks in support of a Web browser's request for service as programmable functions receiving parameters as input and providing as their output handled by the control program agent task completed results for reporting in accordance with the Web browser request in the form and to the location determined by a request and handling these request without needless user intervention.

In accordance with our invention, we have created a way to allow Web users to request information that is created by a data interpretation system (DIS) and then presented by a web server to the user of the web. Our solution provides a way of requesting and processing and presenting information on the Web. In the process, data is retrieved from multiple sources which may be located remotely and accessed via an intranet routing and via the Web Internet and processed by our decision support capsules. Now companies and universities, and other users that want to access data located on different databases, want that data processed and formatted, and presented in a form the user desires, such as a graphical format. Our solution permits users to access information from various sources and obtain information at a desired location as a result of a single request which is responded to by an organization of facilities and command file sub-agent decision support capsule objects by our command program agent. Users of the information can be internal to a company, or external. The result can be furnished to a user at a location which is internal or external to the company, and as specified at a specified location with a form and format desired. This allows a report to be managed by the web support services we provide, and in a form consistent with the request, but without requiring a consistent interface solution.

In order to create a way for Web users to request information generation we provide a web server with a control program agent which is linked to a decision support tool of a data interpretation system server, the application processing agent, and then have that server retrieve, process, and format information which is presented to the user on the Web by the Web server. In our preferred embodiment, we have provided a link between a Hypertext Markup Language (HTML) document using a common gateway interface, and open data interpretation system server (ODAS). As a result, Web clients can request DIS reports to be generated, specify the parameters to be used in generating the reports, and then view the report results on a Web home page. The DIS capsule can generate graphical information, such as colored pie charts, line graphs, bar graphs, and other forms of generated information. Since the Web server is capably of presenting the results in desired formats, the full capabilities of a DIS report are utilized.

Our invention provides a method and system for allowing a user of a client to access and assemble information structured and reported to the user in accordance with his desires, selecting information for disparate servers which are located within a network can be an intranet or internal network, such as a LAN or WAN not normally accessible to the Internet, or coupled to the Internet. In accordance with our invention one can access data on multiple databases of different types using a single user request from a client. We also allow the facility for providing specialized specific requests to be created for routine use, as well as the facility to formulate generalized or specialized ad hoc requests. In addition, we provide besides query and update capability, the ability to perform calculations with respect to any retrieved data, to format the information in text or in graphics, and the facility of presenting the results to the client for display or other use.

The improvements which we have made achieve a means for accepting Web client requests for information, obtaining data from one or more databases which may be located on multiple platforms at different physical locations on an Internet or on the Internet, processing that data into meaningful information, and

presenting that information to the Web client in a text or graphics display as a location specified by the request.

Our invention of providing a web server with a control program agent allows organization of decision support functions to be executed by application processing agent servers located throughout the Internet to gather and supply information not presently available with any existing resources without the need of endless intervention on the part of a requesting user of the WWW; further enabling an ordinary user to take advantage of expertise which is provided by programmable sub-agents developed by those with particular expertise in a given area as well as enabling use of standard routines commonly needed.

These improvements are accomplished by providing for Web clients to request information from an application processing agent in which the application processing agent server performs tasks based on received requests from a client in a distributed environment by a web server supported by an access agent link and control program agent which in turn causes a decision support function to be executed by the application processing agent server. This is performed within the distributed environment by the application processing agent server which forms part of a network coupled to and under control of the control program agent. According to our invention the decision support function is provided by a data interpretation system which functions as part of the application processing agent and the decision support function is programmable and generated by a data interpretation system, DIS or other decision support element performing similar functions, and provided in a form accessible to our control program agent which presents the output generated to be presented to the user on the Web who made the initial request. We have provided, in a preferred embodiment, a link between IBM's Hypertext Markup Language (HTML), the Common Gateway Interface (CGI), and the Open DIS Access Sewer (ODAS), all of which may be used on machines which are are commercially available from IBM. In order to write additional functions which develop our invention, the reader is referred to the Medaphor Data Interpretation System publication "Developing Applications with OpenDIS Access Service, Version 2.0, available from IBM, First Edition (September 1994) Part Number 315-0002-01 which is incorporated herein by reference.

Our improvements relating to our control program agent is in accordance with our preferred embodiment is normally installed on an IBM HTTPD which is an IBM OS/2 Web Server or other server having Hypertext Markup Language and Common Gateway Interface. In our preferred embodiment, the HTTPD incorporates our control program agent and is supported by an access agent which provides the hardware connections to machines on the intranet and access to the Internet, such as TCP/IP couplings. The hardware for the Web sewer is thus a workstation, such as IBM's PS/2 model 80 with OS2. However, the HTTPD can be installed in PCs and upwardly also in machines which range across IBM's line of computers from powerful personal computers to mainframe systems which support MVS, IBM's operating system which enables multiple kinds of operating systems, including "UNIX" to co-exist on a single platform. As a result of our invention Web clients can request DIS reports to be generated by the application processing agent specifying the parameters to be used in generating the reports, and then as a result of the request receive a result which is presented, as a visual display or otherwise, on a Web page for use by the requesting user. Our machine implementation allows a user having DIS access to generate graphical information such as colored pie charts, line graphs, bar graphs, etc. Since Web browsers such as IBM's Web Explorer are capable of displaying these formats, all the functions which can be created by a DIS capsule can be utilized by a user of our invention.

According to our improved method, an Internet World Wide Web user connects to a Web server through the use of a Web browser. In accordance with our preferred embodiment, we use HTML as the language used by Web servers to create and connect documents that are viewed by Web clients. HTML is an example of a hypertext language having the facility of clicking on a highlighted word, string of words, or image in order to move to another HTML document or invoke a program on the server. An example of a Web client would be a machine used by a person using IBM's Web Explorer product. In using our invention a user may click on the hypertext in a document to reference a function which will be provided by an application processing agent server. The user is able to connect

to another document that may be on another Web server. HTML commands are used to reference other documents. HTML is used to reference programs available on a server, and pass parameters to those programs. The application processing agent server executes a program when it is referred to by a Web client via a control program agent resident, preferably, in a Web server.

The Web client selects the information that they wish to view by using the HTML created page, the Web server takes the client request and passes it to a C program implementation of our control program agent. Web servers, such as HTTPD for OS2, with our control program agent are able to provide access to executable programs through the use of the Common Gateway Interface (CGI). When a program is referenced by the HTML, any parameters are passed to the program and it is executed. In our preferred embodiment we have used CGI to invoke programs that we have developed that will interface with the DIS product. CGI is an example of a software gateway from a Web server to programs outside the Web server application.

The control program agent that is called in this instance by the Web server through the CGI interface, passes the Web client request along to a data interpretation system DIS via a Open Dis Access Server (ODAS). ODAS is a feature of a data interpretation system DIS that allows programs to initiate DIS functions, such as invoking DIS capsules. Our control program agents interface with DIS through ODAS to submit DIS capsules for execution. DIS capsules are basically programs that DIS application programmers create with the DIS programming language. In accordance with our invention, we have written capsules which are executed as a DIS capsule on a server to gather data from one or more databases, process that data, and create a report in one of many formats, which we will describe by way of example. After the DIS capsule completes executing, in accordance with our preferred embodiment, the results that are generated during execution of a capsule are stored in a file on the application processing server.

After DIS creates a file that contains the formatted report results, our control program agents program dynamically creates HTML tags to present the formatted report back to the Web client on the Internet. Our control program agents using the CGI interface can create HTML commands dynamically. In this way a program can present information on a Web browser for the Web client.

After the DIS capsule has created the file containing the report request results, the control program creates HTML statements dynamically that display the report results to the Web browser.

Alternative means of presenting the data are shown by alternative routing. The user requesting the report may wish to have the report results sent to another location in addition to or instead of displaying the report results to the Web browser. This information is provided during the request phase. As a result of the alternative report request, and according to the parameters indicated therein, the report results can be sent by the control program via electronic mail, i.e. TCP/IP Sendmail facility and Lotus Notes, to one or more locations on the Internet. The report results can be sent as a file and as a note. The request can request a voice response, which can be routed to a voice response unit. Thus, with a call to a translator, the text can be converted to voice, and even translated along the way. The report results can also be sent to a fax machine, or to a computer that has the capability of receiving fax data.

We use these report concepts to present report files created by DIS capsules on the Web client display.

These and other improvements are set forth in the following detailed description. For a better understanding of the invention with advantages and features, refer to the description and to the drawings.

DRAWING DESCRIPTION:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows schematically an overview of the preferred embodiment and particularly shows a typical distributed computer system which has internal and

external networks including the Internet to connect clients to World Wide Web servers and other servers within the system in which our invention is situate.

FIG. 2 shows a inquiry screen (home page) which is displayed on a client after the client is coupled to its server (which may be an Internet server) by a Web browser.

FIG. 3 is a next screen which illustrates how a request is made according to a users desires, making a request in accordance with our invention with an input screen shown.

FIG. 4 is a sample result screen which is returned to the client after the requested service is provided by the computer system network in accordance with our invention formatted according to the specifications of a DIS capsule.

FIG. 5 is a next screen which illustrates how a request is made according to a users desires, making a request in accordance with our invention by selection from a menu and through the use of image mapping.

FIG. 6 is an example of a graphical result screen which is returned to the client after the requested service is provided by the computer system network in accordance with our invention.

FIG. 7 illustrates a flowchart showing data flow between a web server and decision support system tool such as IBM's Data Interpretation System (DIS), and shows the coupling of a Web client to a Web server and the coupling of a request to execute a DIS capsule and the coupling within the Web server from ODAS to a distributed DIS LAN with heterogeneous connections to multiple databases.

FIG. 8 illustrates as a flow chart the functions of the control program for the web server.

FIG. 9 illustrates by way of example a DIS capsule that creates a text report file.

FIG. 10 illustrates by way of example a DIS capsule that creates a graphical report file.

FIG. 11 illustrates an alternative configuration of the network system as it may be employed for permitting access to information available through homepages and in data warehouses where access to the homepage or database may or may not be restricted by a firewall.

(Note: For convenience of illustration, in the formal drawings FIGURES may be separated in parts and as a convention we place the top of the FIGURE as the first sheet, with subsequent sheets proceeding down and across when viewing the FIGURE, in the event that multiple sheets are used.)

Our detailed description explains the preferred embodiments of our invention, together with advantages and features, by way of example with reference to the following drawings.

DETAILED DESCRIPTION:

DETAILED DESCRIPTION OF THE INVENTION

FIG. 1 illustrates a information delivery solution of a typical combination of resources including clients and servers which may be personal computers or workstations as clients, and workstations to mainframe servers as servers. The various elements are coupled to one another by various networks, including LANs, WANs, and other networks, which may be internal SNA networks or other like internal networks, and also providing access to the Internet, which couples the system to the world via Internet.

The Preferred Embodiment

Turning now to our invention in greater detail, it will be seen from FIG. 1

that our preferred embodiment provides a Web browser 10, which is coupled to a Web server 11. Our Internet WWW browser is an intelligent computer system, such as an IBM PS/2, or other computer, an IBM ThinkPad, an RS/6000 works as well and connections are made to the network via OS/2 WARP Connect, an IBM product. The Internet Web browser in the intelligent computer system which performs the Web browser function has IBM Web Explorer, or NetScape or Mosaic installed thereon. This computer system 10 is bi-directionally coupled with the OS/2 WARP Connect facility over a line or via a wireless system to our preferred computer system which we call our Web server. This system is a PS/2 or RS/6000 or other similar system which includes our control program agent 73, which will be discussed below. Web server 11, in our preferred embodiment is coupled again bi-directionally via a line or wireless coupling to a computer system, such as a PS/2 or RS/6000 or other server which supports and performs the server function of ODAS server 12, which is coupled to the distributed DIS network, here shown as LAN 13. ODAS 12 may be located on the same server as the Web server 11 or be located at a separate service machine, such as an IBM Digital Server. The Web server is logically coupled to our application processing agent server via a network. We call our application processing agent server a DIS File server 14 because is comprises a data interpretation system which supports the decision support functions we provide which is today most inexpensively provided by an IBM computer system which supports OS2. In our preferred embodiment, the intranet network is a LAN. Thus the components of the DIS LAN 13 comprise a DIS File Server 14, a general purpose workstation 15 which can be used for capsule development, a local database server 16, a Capsule Server 17 for storing a plurality of DIS capsules ready for user, a Database Gateway Server 18 which performs the gateway functions to access databases which are linked to it, these databases include geographically distributed databases which can be located, for instance, in Chicago, New York, Dallas, Los Angeles, and each of which can be a different supported database, such as DB2 database 19, ORACLE database 20, Sybase database 21, Redbrick database 22. In our preferred embodiment all servers are coupled with a conventional LAN or WAN connection, with a preferred IBM token ring shown. Reference should also be had to our alternative preferred embodiment discussed below with respect to FIG. 11.

Thus, in connection with the preferred embodiment of FIG. 1 as well as with respect to FIG. 11 it would be appreciated from the schematic overview illustrated by FIG. 1 and FIG. 11 that our invention may be employed in a distributed computer system environment which has internal or intranet networks represented in our preferred embodiment by the DIS Network 13 and external networks including the Internet to connect clients to World Wide Web servers and other servers within the system in which our invention is situate. Our invention makes use of the entire network. The Web browser 10 can make a request to the Web Server 11 for a report. The Web server 11 with the facilities we provide causes the application processing agent which includes our DIS server 14 and its supporting communication server, the database gateway server 18, to act as an agent to gather data from one or more of the multiple databases, including the local database 16, DB2 database 19, ORACLE database 20, Sybase database 21, Redbrick database 22. Further details with respect to the use of our invention for database retrieval of information from multiple databases are provided as to the actions of the application processing agent functions of the database server(s) 18 with reference to FIG. 7.

Thus, returning to our simplified and preferred embodiment, FIG. 2 shows a inquiry screen (home page) 29 in the form which is displayed on a client after the client is coupled to its server (which may be an Internet Web server 11) by a Web browser 10. The entire screen contains information and a plurality of objects. Once the home page is displayed, with appropriate descriptive guidance as illustrated by the FIG. 2, the user can interact, for example, by clicking on image objects 30, 31, 32, 33, 34. As a example should the user want to make a special request in accordance with our invention, he could click on image 30. This would take the user to the next screen, illustrated by FIG. 3. Alternatively the user could select by clicking on image 31 another menu screen, illustrated by FIG. 5. At this point also, a specialized format could be selected by double clicking first on a format select image illustrated by image objects representing access to menu screens 32, 33, 34, one or more of which is a gopher.

The use of selection of icon image object is a function provided by HTML and

programmers knowing this language can readily create variants to the images and functions we have illustrated. Thus incorporated within the drawings are to be understood to be the variants that can thus be created using our examples, as well as extensions and combinations thereof.

When the user selected image 30 by clicking on the image 30, FIG. 3 appears. FIG. 3 is the next screen which illustrates how a request is made according to a users desires, making a request in accordance with our invention with an input screen shown. The content of FIG. 3 is preformatted to except for the user entries which are to be entered in the data input fields 41. In this example the input field 41 is a userid. After a user has entered in field 41 an acceptable input, he would then click on instruction key 42. The instruction key illustrated is submit a request. At this point the Web server captures the information entered by the user, as described in FIG. 7. It will be appreciated that the Web server captures the information entered by the user, including specialized input, as well as any "hidden" default information, which can include password authorizations, charge account identification, and other information that can be used by the system in responding to the request. Thus the system can assume that the "hidden" password is an authorization to perform some function, such as include information from confidential source, or exit to the Internet. The charge authorization can also be tracked and accumulated by the system as it parses through its functions to charge back chargeable usages. If a request is for an order of an item, the actual item requested can be shipped and billed with this information. Since these functions are "hidden" they do not appear in the FIGURE but included with a request. The return of the request is illustrated in FIG. 4.

FIG. 4 is a sample result screen which illustrates how a sample report conforming to the request results are presented to the client after the requested service is provided by the computer system network in accordance with our invention formatted according to the specifications of a DIS capsule which is illustrated by example in FIG. 9. In this example, the return was a file, whose file name is displayed as P81484 at 43. Informative text accompanying the file is included as illustrated by the example information 44. The screen provides the content of file 43 in the requested form of preformatted text 50 in the form of a display of a text report generated by a DIS capsule stored in the DIS server 17. While we show text as the form the report results, the form of the request can be another form of presentation, as and image, a voice response, or other multimedia presentation. Reports can be returned translated into any desired language based upon the request, as may be provided by DIS capsule calls to a translator. These features are included in the result 50 report.

When the user selected image made by clicking on the image 32 in FIG. 2, FIG. 5 appears. FIG. 5 is a next screen which illustrates how a request is made according to a users desires. A user makes a request, in this instance for sales results within the organization for YTD Catalog Revenue in accordance with our invention by entering text data into the data entry areas 41 and 42 of the formatted screen with information as to type of data selected 40A which will be translated into specific report information created by a DIS capsule.

FIG. 6 is a sample result screen which illustrates how the request results are presented to the client after the requested service is provided by the computer system network in accordance with our invention formatted according to the specifications of a DIS capsule. In this instance selection of the object 32 links to the the screen of FIG. 5, which in turn with the DIS capsule created the output shown in FIG. 6. DIS Capsules will be illustrated by examples in FIG. 9 and 10. In this example the output of the DIS capsule illustrated in FIG. 10 is presented on the screen shown by FIG. 6. The screen comprises a file name identifier, descriptive information 61, and preformatted text 60 which is the display of the named file P555119. This is the display of a graphic report showing what might be deemed (but is not) Confidential information relating to Catalog Revenue for 1995 YTD, with revenue given in $M, and breakout as to HDW, SFW, PMV, MN and MNT from selected locations in Chicago, New York, Dallas, and Los Angeles, all of which are located on different systems, and which, as illustrated in FIG. 1, may be on different databases such as DB2, Oracle, and Sybase relational databases. This report was generated by a DIS capsule which is illustrated in FIG. 9. This example illustrates how multiple actions can be taken on information retrieved. In this example data was translated into image

material by calculation and formatting in the form of a graphic pie shaped report. Other image data could also be displayed, as frames of selected images, or a sequence of images in the form of a moving picture display, which can be outputted from a server as will be described in FIG. 11.

FIG. 7 illustrates a flowchart showing data flow between a Web server and decision support system tool such as IBM's Data Interpretation System (DIS). FIG. 7 shows the coupling of a Web client 71 (corresponding to Web browser 10 in FIG. 1) to a Web server 72 (corresponding to Internet WWW server 11) and the coupling of a request to execute a DIS capsule.

The Web browser 71 can make a request to the Web Server 72 for a report through the use of HTML. The HTML document refers to our control program agent 73, which may be implemented with the C language or other language which can provide run code for the particular Web server which is employed. We illustrate our preferred program according to the description provided in FIG. 8. The Web Server 72 passes request data to and invokes our control program 73 through the use of the CGI in accordance with our invention. The control program uses ODAS 74 in ODAS server 12 to set DIS capsule parameters and initiates the execution of a DIS capsule located in this embodiment in DIS capsule server 17 according to our preferred examples illustrated in FIGS. 9 and 10.

After a DIS capsule completes execution, the file created by the DIS capsule contains the formatted report results requested by the user. Our control program 73 dynamically creates the HTML statements that present the file to the Web browser 10 screen. FIG. 7 shows the coupling within the Web server from ODAS 74 to a distributed DIS LAN 75 with heterogeneous connections to multiple databases DB2, Redbrick, Sybase and Oracle. Other sources of data can be linked to the LAN.

Preferred Embodiment Interface between Server and DIS

Our preferred control program agent 73 in FIGS. 1 and 11 is illustrated in detail by way of the flowchart of FIG. 8. In our preferred embodiment, this program can be written in C or other suitable language but for general appreciation of the details, we will describe the steps in detail. These steps can be implemented by programmers of ordinary skill in the art without undue experimentation after understanding the steps described below. The control program agent 73 is located in a Web server and provides an interface and execution functions. Thus in FIG. 11 the function is provided between the Web Server 131 (corresponding to Internet WWW server 11 in FIG. 1) and DIS which is located in a DIS server 133 (corresponding to server 14 in FIG. 1) and for presentation of results according to the instructions of the Web browser 130 (corresponding to browser 10 in FIG. 1) according to the request command, which in default is a return to the Web browser home page. This interface utilizes in our preferred embodiment the Web Server CGI and the DIS ODAS.

Before we proceed to the control program 73, it will be noted that in FIG. 11 the Web Browser 130 will link to a Web Server 131 accessing it on the Internet though a unique ID called the uniform resource locater to access the node which we call the Web server 131. When that access takes place an HTML document is displayed by the Web server 131 to the Web browser 130, as shown in FIG. 2. Now, the user makes his entries as described with respect to FIG. 2. Next the HTML document refers to the control program agent 73 and the Web server 131 through the use of the CGI invokes our control program agent 73. The Web server 131 retrieves data entered by the user from the HTML document and passes that data to our control program agent 73 upon invocation.

The Web Server 131 has a gateway interface that allows the server to invoke a control program agent 73 running on it and to pass input parameters to the control program agent 73 (FIG. 8) that were returned from the Hypertext document of the Web Browser. It will be appreciated that while we illustrate for our preferred example a single Web Server 131, the Hypertext document locates the particular Web Server that can support the request made by checking the details of the "hidden" defaults and those functions requested. Thus a menu request for a generalized search throughout the Internet may locate the particular service machine having an application processing agent which has the information desired. Once the control program 73 (FIG. 8) is invoked, the steps programmed for the machine to follow begins with a step 110 illustrated in FIG.

8. In reviewing this preferred control program agent it should be appreciate that steps 110 and step 111 are steps that are interchangeable in order and which obtain environment variable data from the HTML document return.

Thus step 110 obtains a PATH.sub.-- INFO environment variable data. PATH.sub.-- INFO contains data from the HTML document that referred the Web Server to our program. Specifically the data contains the name of the DIS capsule to call, the name of the file containing the HTML statements to use when building the HTML document that displays the DIS capsule results to the Web browser, and the type of file that the DIS capsule will create. All off this information is the variable data which is stored in a buffer environment in step 112, and which is used in subsequent steps.

Thus also, the control program proceeds with step 111 which may follow or precede or proceed in parallel with step 110 to obtain the QUERY.sub.-- STRING environment variable data. QUERY.sub.-- STRING contains data from the HTML document that referred the Web Server to our program. Specifically the data contains values selected by the user and/or default values selected by the HTML document designer. These values are set in the DIS capsule by our control program prior to execution of the DIS capsule. This information is used to set variables in the DIS capsule. All off this information is the variable data which is stored in a buffer environment in step 112, and which is used in subsequent steps.

Within the scope of the discussion of the control program agent illustrated by FIG. 8 it should be appreciate that the steps 112 through 125 include the utilization of an API set that provides a method of invoking executable programs located in a service machine which we denote as a sub-agent which executes in step 122 object capsules from our sub-agent DIS file server 14. This provides functions such as queue and update functions for databases on multiple platforms and allows the processing of data retried from a database to be performed, including executing calculations, doing formatting, charging of accounts and the storing of results as a file accessible to the control program agent. During processing our control program agent 73 provides setups for API calls which occurs in steps WHAT ARE THESE STEPS. Thus the control program agent will proceed as with an API set with step 113.

With the variable information now stored in a buffer, in step 113 the control program retrieves from a store, all of the DIS capsules that are used and the variable names associated with each DIS capsule and loads into memory associated with the control program the DIS capsule names available and the variable names associated with each DIS capsule.

At that point in step 114 the control program is ready to and does initialize a connection between our control program and the ODAS through the use of an ODAS API. In other environments another API performing similar functions could be used.

At that point, if required for control by the decision support system, and as required by DIS, the control program would log onto the port or desktop for the assigned user. Thus, our control program agent 73 in step 115 logs onto a DIS "desktop", our DIS file server 14.

Once the DIS capsule information is loaded into control program memory, the control program can and does in step 116 retrieve from its memory the DIS capsule variable names associated with the DIS capsule name passed to our control program in the step 110 where PATH.sub.-- INFO is provided.

Next, in step 117 the control program creates a data array stored in the control program memory containing the capsule variable names and the values for them that were Passed to our control program in the QUERY.sub.-- STRING step. These two steps 116 and 117 should be done in order, even though steps 110 and 111 can have an arbitrary order. At this point in step 117 you are matching the DIS capsule variable names with the data that was passed to the control program in the QUERY.sub.-- STING environment variables.

Next, in preparation for a report, in step 118 the program creates a unique filename which may include data originated by the HTML document's variables stored in step 112 (dotted line) to pass to the DIS capsule as a DIS variable

for use in naming the report which will be created by the DIS capsule. As a result, the DIS capsule will create that file with the unique file name during its process.

In anticipation of DIS capsule execution, the values of variables used by the DIS capsule are obtained from the data array in the control program memory containing the DIS capsule variable names and the values for them that were passed to our control program in the QUERY.sub.-- STRING step. This is done in step 119 using the ODAS API to set the DIS capsule variable values.

At this point the capsule server 17 for the DIS server 133 attached to the Web Server 131 via network 132 will have a DIS capsule services queue. This queue is the queue of jobs being requested of the Dis Capsule Server 17. For the current job request (other like requests being perhaps still in the queue) we use the ODAS API to query the contents of the DIS Capsule Services queue. If the queue size is larger (>t) than a threshold level, then the process enters a wait state until the queue size is reduced to a tolerable level. The queue test of step 120 is a loop test which returns to test the queue size until a test answering "is the queue of a size that execution can proceed?" (Whenever the queue test is answered YES, at that point the ODAS API is used to submit a DIS capsule for execution in step 121.

After the ODAS API submits a DIS capsule for execution the particular request process being executed by the control program enters a wait state until completion of the DIS capsule execution. For this step of the process the control program uses the ODAS API to wait for completion of the DIS capsule execution performed by the DIS capsule execution 122. During a wait state other requests can be processed by the control program, as requests are fed through the control program as a pipeline, in this WAIT PIPE API step 123, so that the control program continually advances requests through the system.

During the wait state 123 the ODAS API looks for a completion signal. When that is received, the control program then in step 124 reads the file identified by the name passed to the control program in the first PATH.sub.-- INFO step that contains the HTML statements which are to be presented with the DIS report results.

While in step 124 the control program reads the file identified, it dynamically creates new HTML statements to display the preformatted text to the Web browser. The new HTML statement include the information retrieved from the file in step 113 so that it can be displayed as a header 44 accompanying the report to be displayed, along with the filename 43.

At this point, in step 125 the control program tests for the kind of report to be created by obtaining information from stored variables and identifies output parameters, such as whether the report is to be a text report, or a graphical report. At this point the control program branches to the sequence applicable to the kind of report to be created. If the output is to be routed the the Web server 10, then the output is routed to the Web server in step 126.

If a text file report is created by the DIS capsule, that determines that a text display is to be reported and the the control program reads the file created by the DIS capsule and dynamically creates HTML statements to display the data lines to the Web browser.

If a graphics file is created by the DIS capsule, that determines that a graphics display is to be reported and the the control program dynamically creates the HTML statement to display the graphics file to the Web browser.

On the other hand, the control program agent allows alternative output direction, and if the output is another type, or an additional output, as for broadcast, it can be routed to another destination. In step 127, we illustrate how using the IBM Digital Server, output can be routed to a requestor selected resulting output selected from a group of possible output units, comprising fax, printer, retail or banking installations, or provided as a series of full motion videos or still frames which are can be transmitted to display devices, such as a TV set under control of end users with a set-top box cable control. These facilities are provided by providing the output of our control program agent from the web server to the alternative output device 127, In this case,

the IBM Digital Server, which with an RS/6000 CPU, Network I/F Bus, DISKs, modems, and X.25 Data Switch provides the hardware to route the output to a variety of output devices, to fax, printer, retail, banking, TV or cable customers via the digital server service machine for full motion and still video, supplied with MPEG 2 and MPEG 1 protocol images respectively, to subscribers,

Along the way, the output can be coupled to an auxiliary function, such as back-up or accounting processes 128 which allow for charging for system utilization and service charges for services and items requested. These processes will make use of hidden variables associated with the request, such as charge authorization. One of the hidden variables which may be associated with a request is a credit card number. The credit card number, is preferably encrypted, with a DES or RSA encryption utility, and this along with access authorization variables, will allow access to sensitive databases which reside behind firewalls. If selected data according to the request is permitted to the access authorized user at the location inside or outside the Internet, the data can be included in the results reported by our system to the Web browser.

Preferred Embodiment of Text DIS Capsule

In accordance with our invention, an HTML document, which is running on a web server, refers to the control program agent. The web server then invokes the control program agent. The control program agent has access to command files, which provide the preferred file command objects in the form of DIS capsule objects, or DIS capsules as they are known. The command file contains a list of available DIS capsules. Accordingly, there is not need for the HTML document to know how to get to the command file, as the control program supplies this access. A capsule object, as a DIS capsule, can call other routines which may be written in well known programming languages such as Visual Basic or C. These routines become part of the capsule object by the reference, and these routines perform such functions as account tracking, compression, calculation, handling specific custom outputs such as video, voice, translation, and enable programmability of the capsule objects. The capsule objects also have standard object capability, and we will illustrate these by way of the specific examples described.

It will be seen that the control program 73, described in detail in FIG. 8 acts in concert with DIS capsule execution. The DIS capsule is an object program with executable additions which we have created to interact with the control program. It should be also understood that the DIS capsule object can perform programmable functions on data which is retrieved from databases. Not only can a DIS capsule get data, it can combine, reformat, and update, the data retrieved. It can act on the data to create new data and basically act as a dedicated processor processing data gathered or created during a Web browser request to output the end result to the user under programmable parameters determined by the creator of the DIS capsule, as they may be selected, if desired, by the user as pad of the request. Thus the user entered inputs as part of his request, either free form or by selection of variables in the menus afforded to the user as illustrated by way of example in FIG. 5.

DIS capsule objects are like some other objects. For instance in Microsoft's products, an example being the Excel (trademark of Microsoft) spreadsheet, one can click on an object portrayed on the screen and link a succession of objects to perform a specific function, such as take data from a spreadsheet and reformat it into a variety of selectable formats, such as text or graphic illustration. The kind of action to be taken is illustrated by an object on the screen, and linking of routines is done by a succession of clicks on icons representing the object.

In accordance with our preferred embodiment, a DIS capsule is used to invoke system resources. This is done by providing a list of commands, which may be those provided by a DIS processor itself, or written in Visual Basic or C by the programmer. The result is a command file, like an exec or command file in OS/2 or like a *.BAT file in DOS. These capsules perform the specific functions that are requested by the user from his initiation session. The user further qualifies the execution of the DIS capsule by providing parameters which are used in the invocation.

Now the DIS server 133 supports DIS, the program processor which supports DIS capsules by processing commands contained in the DIS capsule, either directly, in the case of DIS functions, or by to other system or user supplied functions. The user supplied functions comprise mainly those DIS functions which are supplied by DIS and illustrated in the manual "Developing Applications with OpenDIS Access Service, Version 2.0 of the OPEN Access Service." For those not familiar with command files, this manual is fully incorporated herein by this reference as available at the USPTO. An example of a system supplied function would be the base support for SQL queries of a specific database, which are invoked by the DIS capsule program.

In illustrating the specific examples of our invention illustrated in FIGS. 9 and 10, both illustrate linked objects according to a specified flow sequence within a DIS Environment. The DIS environment contains numerous functions, including the Internetwork routing functions which the DIS capsules can invoke. Thus, a DIS object which queries a database, as illustrated, invokes the Internetwork routing functions to query databases where they are located on the network. If the preferred example of DIS environment is not supplied, a similar environment with program environment means which supports reaching a destination on the Internet by a link between systems which routes data from one physical unit to another according to the applicable protocol should be supplied. The protocol will employ a URL address for Internet locations.

FIG. 9 illustrates by way of example a DIS capsule that creates a text report file. Referring to FIG. 9, it will be seen that the capsule, represented by a series of linked objects, is supported by Internetwork processor support environment means 90. Within this environment an integrated capsule creates a text report file as a result of the object 95, make text. This object result file is the file 43 according to FIG. 3 which is displayed at the browser. In the illustrated example, the multiple DIS capsule data retrieval command file 91(a) . . . 91(n) initiates as a first step multiple queries to different databases which are specified by the parameters of the request. In this example, multiple queries are initiates as SQL type search requests as multiple steps 91(a) . . . 91(n) executed by the DIS capsule server 33 with the Database Gateway 134 to select data from DB26000 databases located inside the intranet 140 and on the Internet by Internetwork routing to database gateway 134' and its DB26000 databases by step 91(a). The data is stored in a DIS declared buffer. Similarly, in parallel or successively, additional steps 91(b), 91(c), 91(d), and 91(n) retrieve data and store in their object buffer data retrieved from Sybase, Oracle, Redbrick, and IBM's Data Warehouse databases. Thus object 91(a) will query DB26000 and bring data back to DIS. Object 91(b) will query Oracle and bring data back to DIS. Object 91(c) will query Sybase and bring data back to DIS. Object 91(d) (shown as a dot in FIG. 9) will query Red brick and bring data back to DIS, and so on. The nth object 91(n) will query IBM's data warehouse and bring data back to DIS. In a subsequent linked processing step 92 data from the database queries in the first step is joined by joining object command file 92 and stored in a buffer related to this object. Object 92 will join the data from the n locations searched in step 91. Thereafter, in a subsequent processing step performed by calculation object command file 93 on the joined data in the joined database result buffer of step 92, desired calculations performed in accordance with the parameters indicated by the request are done on the joined data. Thereafter, in accordance with the request parameters text is formatted to space delimited text by the format object command file 94. The results are stored in a buffer associated with format object command file 94. Thereafter, a make text command file 95 causes the formatted text to be created as a text file for the WWW sewer 131 to be stored in a file which is accessible to and can be retrieved and displayed by the control program agent 73, or directly displayed by the control program agent 73 in the form illustrated in FIG. 4 at the Web browser 130. It will be noted we have illustrated this process as object capsules in a DIS internetworking environment. These object capsules are a specialized form of a command file, which can encompass additional commands called by an object.

Preferred Embodiment of Graphics DIS Capsule

FIG. 10 illustrates by way of example a DIS capsule that creates a graphical report file. For simplicity, data in this FIGURE is also shown in a DIS environment 90. Retrieval object command file 101 illustrates a step of retrieval of data from one or more databases as specified in the parameters of

the request, performing these retrieval steps as did retrieval object command files 91(a) . . . 91(n). Thereafter, this data is plotted with the make plot object command file 102, with the results being stored in a buffer. The final step of creating a result- to-be-presented file, in this instance in the form of a bitmap ready for display to a Web browser 130 is created by the make bitmap (BMP) object command file 103. The example of a preferred bitmap object command which would be employed with todays Internet environment is a GIF image. Others can be used as well. Again the results are provided to the Web browser 130, by the action of the program command agent 73 on the Web Server 131, the results being illustrated by the pie-chart of FIG. 6 in accordance with the parameters of the request for generating the graphical report illustrated by FIG. 6.

Alternative Preferred Embodiments

FIG. 11 illustrates an alternative configuration of the network system as it may be employed for permitting access to information available through homepages and in data warehouses where access to the homepage or database may or may not be restricted by a firewall. In FIG. 11, the web browser(s) 130 accesses an associated Web Server 131,131', 131" either by a coupling or addressing with a uniform resource locator (URL) the Web Server 131 which may be selected with a Hyperlink. This can be a direct coupling or an indirect coupling, as via a node locatable in a common access medium, such as provided by Internet resources accessible via a web browser, e.g. supporting Web Explorer, or Mosaic, NetScape, node 131 located somewhere on the Internet which utilizes our control program agent 73. Now node 131 which functions as a Web server is coupled via a token-ring network, SNA network, or other suitable network 132 (one of the any which may be used on the Internet as a transmission medium) with the facilities provided within what we will call our intranet, those facilities which are "proprietary" to the owner and which may be protected by firewalls at the intranet boundary 140. Now note that our control program 73 is resident within the Web Server 131 and functions as described in FIG. 8 to couple to a DIS server 133 located within the intranet 140, which is preferably located behind a firewall as indicated in FIG. 11. This DIS Server 133 is in turn coupled to our Database gateway 134. This database gateway is configured as illustrated also in FIG. 1 for gathering information from databases coupled to it and located on servers for DB2, Oracle, Sybase, and Redbrick, as well as one for information warehouse functions. In our preferred embodiments these database units are IBM mainframe systems, as available commercially today, but they could be AS400s, RISC/6000, RISC/6000 SP or other systems supporting the databases.

The DIS Server is a server which supports DIS or similar decision support functions and the functions provided by our DIS capsules illustrated by FIG. 9 and 10.

Now our Web browsers 130 can not only access information within the intranet, but can reach outside the intranet to gather information located elsewhere via the Internet. We will describe two examples of our preferred couplings to elements on the Internet. One example couples the database gateway 134 to another (a second) database gateway 134' via the Internet and it Internetwork routing (INR) protocol available from IBM as part of its current DIS product which can make use of UALs. The second database gateway 134' is coupled to its own (second) DIS server 133'. At this point the Web browser 130 can access data not only intranet, but also via the Internet to gather data from a database supported by DIS server 133' located outside the intranet. The Database server 134' would be able to gather information from any database coupled to it, as illustrated, assuming access is public or accessible after processing of a hidden variable access authorization.

However, the web browser(s) 130 can also access via Web Server 131 (with our control program 73 illustrated in detail in FIG. 8) another Web server 131' which implements our control program 73. This Web server, for example, Web server 131' can also be coupled via its own (second) network 132' (which supports functions equivalent to network 132 and as illustrated in FIGS. 1 and 11) to an associated DIS Server 133' as illustrated to perform tasks like those we are describing from a request sent via the second network from its Web server 131'.

However, as another alternative example, Web server 131' with an appropriate API can access a directly coupled database available to the server, such as MicroSoft's Access 131a. Thus small databases which have not yet invested in being able to gather information from an intranet resource, can use their own direct resources, and also be interrogated by the Web browser(s) 130, or another web browser 136. Remember that browser's 130 can also communicate with the Web server 131' across the Internet, just as can a Web browser 136 located on the intranet 140 inside of the firewall illustrated by the intranet 140 dashed line shown in FIG. 11. With a browser 136 in place at the Web Server 131' location, that browser 136 can make requests, if authorized across the intranet to the Web Server 131 which can then utilize the DIS capsules provided by the DIS Server 133.

Physically, the network 132 will have its own access server 135 preferably in the form of a TCP/IP server 135 to make the physical connection across the Internet. We illustrate in FIG. 11 this other logical layer as located on the network. This TCP/IP server supports the physical connections which are needed by the other logical higher levels of service supported on the network. The use of an InterNetwork Routing Protocol (INR) allows the logical coupling illustrated between a application processing server 134 to an external intranet application processing server 134'. On each network there can be one or more web servers. A Hypertext document request asking for a field to be seached, as by a Hyperlink, could index to a server directly, e.g. a second web server 134" on the same network which would have its own control program agent function duplicating the control program agent resident in web server 134. Thus at the request homepage a menu which say if "Art&Literature search", when selected in a Hyperlink setting, would index to a particular web server and a particular document within that web server's environment. This web server 134" besides being linked to its own application processing server 133" has a direct link, in the environment illustrated, to an MVS CICS, a transaction processing server for handling transaction processing. Such a solution allows CICS transaction processing to utilize the Internet to save transmission costs and still be located beneath a firewall for retention of data integrity. The outputs provided by the web server to the requested destination can be outside of the firewall, and in the form of results illustrated by the possible examples shown in FIGS. 3, 5 and 8.

While we have described our preferred embodiments of our invention, it will be understood that those skilled in the art, both now and in the future, may make various improvements and enhancements which fall within the scope of the claims which follow. These claims should be construed to maintain the proper protection for the invention first disclosed.

CLAIMS:

What is claimed is:

1. A computer network comprising a plurality of servers, each supporting at least one client computer, said network comprising:

said client computer for making requests;

said server coupled to said client for receiving and fulfilling a request as an agent of said client;

a plurality of information access servers, for acting a sub-agents for said server during a process of fulfilling requests,

said information access servers providing access to capsule objects which perform programmable functions which are executable upon a received command initiated from said server,

said server including a control program agent for receiving a user request initiated at the client computer for information and for transmitting said request to a sub-agent information access server having capsule objects which execute upon control programmable functions requested by said server;

said sub-agent information access servers being coupled directly and/or via the

network to a plurality of database resource gateways for information retrieval from ones of a plurality of database resources having data which may fulfill a data need of said request;

said sub-agent information access servers executing a capsule object to cause any relevant information contained in said plurality of database resources which fulfill a data need of said request to be retrieved and processed by said sub-agent capsule object,

said sub-agent after retrieval from the databases and processing of said data storing said retrieved and processed data as results in a file created for return to said control program agent of said server and returning said created file to said server in response to said control program agent transmission,

said control program agent of said server upon receipt of said file from said sub-agent causing a report of said results of said sub-agent's processed to a facility determined by said client request; and wherein said network includes

a web browser, means for associating said web browser with a homepage including

a first control program agent node supporting a control program agent coupled to and supporting said homepage and supporting an API to access a database available to said first control program agent node,

said control program agent and API enabling a user of said web browser to gather information from said database available to said first control program agent node and to gather information from an intranet resource and to provide access thereto in response to an interrogation initiated at a remote web browser, and

wherein said web browser is at a web server location with said web server providing said control program agent node, and browser requests, if authorized for access across said intranet, accesses a command file agent in a web server on said intranet providing said second command file agent node, which then utilize capsule objects provided by a server functioning as a command file server.

2. A computer network comprising a plurality of servers, each supporting at least one client computer, said network comprising:

said client computer for making requests;

said server coupled to said client for receiving and fulfilling a request as an agent of said client;

a plurality of information access servers, for acting a sub-agents for said server during a process of fulfilling requests,

said information access servers providing access to capsule objects which perform programmable functions which are executable upon a received command initiated from said server,

said server including a control program agent for receiving a user request initiated at the client computer for information and for transmitting said request to a sub-agent information access server having capsule objects which execute upon control programmable functions requested by said server;

said sub-agent information access servers being coupled directly and/or via the network to a plurality of database resource gateways for information retrieval from ones of a plurality of database resources having data which may fulfill a data need of said request;

said sub-agent information access servers executing a capsule object to cause any relevant information contained in said plurality of database resources which fulfill a data need of said request to be retrieved and processed by said sub-agent capsule object,

said sub-agent after retrieval from the databases and processing of said data

storing said retrieved and processed data as results in a file created for return to said control program agent of said server and returning said created file to said server in response to said control program agent transmission,

said control program agent of said server upon receipt of said file from said sub-agent causing a report of said results of said sub-agent's processed to a facility determined by said client request; and wherein said network includes

a web browser at said client computer for making requests,

means for associating said web browser with a homepage at said server by a coupling or addressing with a uniform resource locator,

a control program agent at said server node located somewhere on the Internet supporting said control program agent coupled to and supporting said homepage by a coupling or addressing with a uniform resource locator,

said control program agent server being coupled via a network with facilities provided within an intranet for private owner facilities and which may be protected by firewalls at the intranet boundary,

said control program agent being coupled to an information access server functioning as a command file server and said command file server being coupled to a database gateway for gathering information from databases coupled to said database gateway and located on different database servers, said command file server supporting a plurality of command file objects which are programmed to perform web browser service support functions at the request of a user of said web browser to access information within the intranet and to gather information located elsewhere via the Internet as a sub-agent of said control program agent; and

wherein by submission of a request at a web browser a user can not only access information within an intranet, but can reach outside the intranet to gather information located elsewhere via the Internet.

3. A computer network according to claim 2

wherein said server which is said server coupled to said client for receiving and fulfilling a request as an agent of said client is a web server for supporting a web browser,

and said server includes

means for receiving from a world wide web browser a request to be fulfilled as an agent of the browser client, and

a control program agent for organizing organizing distributed sub-agents as distributed integration solution servers on an intranet network supporting the web server which also has an access agent servers accessible over the Internet.

4. A computer network according to claim 3, further comprising

a plurality of distributed integration solution servers for executing selected capsule objects which perform programmable functions upon a received command from said web server control program agent.

5. A computer network according to claim 4, further comprising

a database gateway coupled to a plurality of database resources for supplying upon a single request made from a Hypertext document, requested information from multiple data bases located at different types of databases geographically dispersed.

6. A computer network according to claim 4, further comprising

command objects for performing calculations, formatting, and other services prior to reporting to the web browser or to other locations, in a selected format a requested result report selected from a set of result reports,

including a display report, facsimile report, a printer report, a report to customer installations, and a report to TV video subscribers, with account tracking.

7. A computer network comprising a plurality of servers, each supporting at least one client computer, said network comprising:

said client computer for making requests;

said server coupled to said client for receiving and fulfilling a request as an agent of said client;

a plurality of information access servers, for acting a sub-agents for said server during a process of fulfilling requests,

said information access servers providing access to capsule objects which perform programmable functions which are executable upon a received command initiated from said server,

said server including a control program agent for receiving a user request initiated at the client computer for information and for transmitting said request to a sub-agent information access server having capsule objects which execute upon control programmable functions requested by said server;

said sub-agent information access servers being coupled directly and/or via the network to a plurality of database resource gateways for information retrieval from ones of a plurality of database resources having data which may fulfill a data need of said request;

said sub-agent information access servers executing a capsule object to cause any relevant information contained in said plurality of database resources which fulfill a data need of said request to be retrieved and processed by said sub-agent capsule object,

said sub-agent after retrieval from the databases and processing of said data storing said retrieved and processed data as results in a file created for return to said control program agent of said server and returning said created file to said server in response to said control program agent transmission,

said control program agent of said server upon receipt of said file from said sub-agent causing a report of said results of said sub-agent's processed to a facility determined by said client request; and wherein said network includes

a web browser at said client computer for making requests,

means for associating said web browser with a homepage at said server by a coupling or addressing with a uniform resource locator,

a control program agent at said server node located somewhere on the Internet supporting said control program agent coupled to and supporting said homepage by a coupling or addressing with a uniform resource locator,

said control program agent server being coupled via a network with facilities provided Within an intranet for private owner facilities and which may be protected by firewalls at the intranet boundary,

said control program agent being coupled to an information access server functioning as a command file server and said command file server being coupled to a database gateway for gathering information from databases coupled to said database gateway and located on different database servers, said command file server supporting a plurality of command file objects which are programmed to perform web browser service support functions at the request of a user of said web browser to access information within the intranet and to gather information located elsewhere via the Internet as a sub-agent of said control program agent; and

wherein there are on the network including an intranet and the Internet a plurality of database gateways, and at the command of a command file running within a command file server one database gateway is coupled to another

database gateway via the network by an inter-network routing protocol.

8. A computer network according to claim 7,

wherein a web browser initiated request is distributed via an intranet to the Intranet whereby access of data is obtained not only intranet, but also via the Internet to gather data from a database supported by a command file server located outside the intranet.

9. A computer network according to claim 7,

wherein a web browser initiated request is distributed via an intranet to the Intranet whereby access of data is obtained not only intranet, but also via the Internet to gather data from a database supported by a command file server located outside the intranet and coupled to said command file server with public access or access obtained after processing of variable access authorization data provided through said command file server.

10. A computer network according to claim 7, further comprising

wherein by submission of a request at a web browser a user can not only access information within an intranet, but can reach outside the intranet to gather information located elsewhere via the Internet.

11. A computer network according to claim 10

wherein said server which is said server coupled to said client for receiving and fulfilling a request as an agent of said client is a web server for supporting a web browser,

and said server includes means for receiving from a world wide web browser a request to be fulfilled as an agent of the browser client,

a control program agent for organizing organizing distributed sub-agents as distributed integration solution servers on an intranet network supporting the web server which also has an access agent servers accessible over the Internet.

12. A computer network according to claim 11, further comprising

a plurality of distributed integration solution servers for executing selected capsule objects which perform programmable functions upon a received command from a web server control program agent.

13. A computer network according to claim 11, further comprising

a database gateway coupled to a plurality of database resources for supplying upon a single request made from a Hypertext document, requested information from multiple data bases located at different types of databases geographically dispersed.

14. A computer network according to claim 11, further comprising

command objects for performing calculations, formatting, and other services prior to reporting to the web browser or to other locations, in a selected format a requested result report selected from a set of result reports, including a display report, facsimile report, a printer report, a report to customer installations, and a report to TV video subscribers, with account tracking.

15. A computer network comprising a plurality of servers, each supporting at least one client computer, said network comprising:

said client computer for making requests;

said server coupled to said client for receiving and fulfilling a request as an agent of said client;

a plurality of information access servers, for acting a sub-agents for said

server during a process of fulfilling requests,

said information access servers providing access to capsule objects which perform programmable functions which are executable upon a received command initiated from said server,

said server including a control program agent for receiving a user request initiated at the client computer for information and for transmitting said request to a sub-agent information access server having capsule objects which execute upon control programmable functions requested by said server;

said sub-agent information access servers being coupled directly and/or via the network to a plurality of database resource gateways for information retrieval from ones of a plurality of database resources having data which may fulfill a data need of said request;

said sub-agent information access servers executing a capsule object to cause any relevant information contained in said plurality of database resources which fulfill a data need of said request to be retrieved and processed by said sub-agent capsule object,

said sub-agent after retrieval from the databases and processing of said data storing said retrieved and processed data as results in a file created for return to said control program agent of said server and returning said created file to said server in response to said control program agent transmission,

said control program agent of said server upon receipt of said file from said sub-agent causing a report of said results of said sub-agent's processed to a facility determined by said client request; and wherein said network includes

a web browser at said client computer for making requests,

means for associating said web browser with a homepage at said server by a coupling or addressing with a uniform resource locator,

a control program agent at said server node located somewhere on the Internet supporting said control program agent coupled to and supporting said homepage by a coupling or addressing with a uniform resource locator,

said control program agent server being coupled via a network with facilities provided within an intranet for private owner facilities and which may be protected by firewalls at the intranet boundary,

said control program agent being coupled to an information access server functioning as a command file server and said command file server being coupled to a database gateway for gathering information from databases coupled to said database gateway and located on different database servers, said command file server supporting a plurality of command file objects which are programmed to perform web browser service support functions at the request of a user of said web browser to access information within the intranet and to gather information located elsewhere via the internet as a sub-agent of said control program agent; and

wherein there are on the networking including an intranet and the Internet a plurality of database gateways, and at the command of a command file running within a command file server one database gateway is coupled to another database gateway via the network by an inter-network routing protocol invoking coupling of database gateways by UALs.

16. A computer network according to claim 15, further comprising

wherein by submission of a request at a web browser a user can not only access information within an intranet, but can reach outside the intranet to gather information located elsewhere via the Internet.

17. A computer network according to claim 16

wherein said server which is said server coupled to said client for receiving and fulfilling a request as an agent of said client is a web server for

supporting a web browser,

and said server includes means for receiving from a world wide web browser a request to be fulfilled as an agent of the browser client,

a control program agent for organizing organizing distributed sub-agents as distributed integration solution servers on an intranet network supporting the web server which also has an access agent servers accessible over the Internet.

18. A computer network according to claim 17, further comprising

command objects for performing calculations, formatting, and other services prior to reporting to the web browser or to other locations, in a selected format a requested result report selected from a set of result reports, including a display report, facsimile report, a printer report, a report to customer installations, and a report to TV video subscribers, with account tracking.

19. A computer network according to claim 16, further comprising

a plurality of distributed integration solution servers for executing selected capsule objects which perform programmable functions upon a received command from a web server control program agent.

20. A computer network according to claim 16, further comprising

a database gateway coupled to a plurality of database resources for supplying upon a single request made from a Hypertext document, requested information from multiple data bases located at different types of databases geographically dispersed.

21. A computer network comprising a plurality of servers, each supporting at least one client computer, said network comprising:

said client computer for making requests;

said server coupled to said client for receiving and fulfilling a request as an agent of said client;

a plurality of information access servers, for acting a sub-agents for said server during a process of fulfilling requests,

said information access servers providing access to capsule objects which perform programmable functions which are executable upon a received command initiated from said server,

said server including a control program agent for receiving a user request initiated at the client computer for information and for transmitting said request to a sub-agent information access server having capsule objects which execute upon control programmable functions requested by said server;

said sub-agent information access servers being coupled directly and/or via the network to a plurality of database resource gateways for information retrieval from ones of a plurality of database resources having data which may fulfill a data need of said request;

said sub-agent information access servers executing a capsule object to cause any relevant information contained in said plurality of database resources which fulfill a data need of said request to be retrieved and processed by said sub-agent capsule object,

said sub-agent after retrieval from the databases and processing of said data storing said retrieved and processed data as results in a file created for return to said control program agent of said server and returning said created file to said server in response to said control program agent transmission,

said control program agent of said server upon, receipt of said file from said sub-agent causing a report of said results of said sub-agent's processed to a

facility determined by said client request; and wherein said network includes

a web browser at said client computer for making requests,

means for associating said web browser with a homepage at said server by a coupling or addressing with a uniform resource locator,

a first control program agent at said server located somewhere on the Internet supporting said a control program agent coupled to and supporting said homepage by a coupling or addressing with a uniform resource locator,

said second control program agent node being coupled via a network with facilities provided within an intranet for private owner facilities and which may be protected by firewalls at the intranet boundary,

a second control program agent node located somewhere on the Internet supporting a second control program agent by a coupling or addressing with a uniform resource locator,

said second control program agent node being coupled via a network with facilities provided within an intranet for private owner facilities and which may be protected by firewalls at the intranet boundary,

said first control program agent being coupled to said second control program agent node located somewhere on the Internet supporting said second control program agent and coupled to and supporting a a command file server, said command file server being coupled to a database gateway for gathering information from databases coupled to said database gateway and located on different database servers, said command file server supporting a plurality of command file objects which are programmed to perform web browser service support functions at the request of a user of said web browser to access information within the intranet and to gather information located elsewhere via the Internet as a sub-agent of said control program agent.

22. A computer network according to claim 21,

wherein said first control program agent resides on a first web server supporting said web browser and said second control program agent resides on a second web server which is coupled via its own network to an associated command file server to perform tasks requested by said web browser and communicated to said web browser after passing through multiple networks.

23. A computer network comprising a plurality of servers, each supporting at least one client computer, said network comprising:

said client computer for making requests;

said server coupled to said client for receiving and fulfilling a request as an agent of said client;

a plurality of information access servers, for acting a sub-agents for said server during a process of fulfilling requests,

said information access servers providing access to capsule objects which perform programmable functions which are executable upon a received command initiated from said server,

said server including a control program agent for receiving a user request initiated at the client computer for information and for transmitting said request to a sub-agent information access server having capsule objects which execute upon control programmable functions requested by said server;

said sub-agent information access servers being coupled directly and/or via the network to a plurality of database resource gateways for information retrieval from ones of a plurality of database resources having data which may fulfill a data need of said request;

said sub-agent information access servers executing a capsule object to cause any relevant information contained in said plurality of database resources

which fulfill a data need of said request to be retrieved and processed by said sub-agent capsule object,

said sub-agent after retrieval from the databases and processing of said data storing said retrieved and processed data as results in a file created for return to said control program agent of said server and returning said created file to said server in response to said control program agent transmission,

said control program agent of said server upon receipt of said file from said sub-agent causing a report of said results of said sub-agent's processed to a facility determined by said client request; and wherein said network includes

a web browser, means for associating said web browser with a homepage including

a first control program agent node supporting a control program agent coupled to and supporting said homepage and supporting an API to access a database available to said first control program agent node,

said control program agent and API enabling a user of said web browser to gather information from said database available to said first control program agent node and to gather information from an intranet resource and to provide access thereto in response to an interrogation initiated at a remote web browser.

24. A computer network according to claim 23,

wherein said remote web browser is also coupled to a second control program agent node located on the Internet, said second control program agent node supporting a second control program agent supporting an API to access a database available to said first control program agent node via said second control program agent

said second control program agent and API enabling a user of said web browser to gather information from a database available to said first control program agent node via said second control program agent node and to gather information from an intranet resource and to provide access thereto in response to an interrogation initiated at said web browser across the Internet by a coupling or addressing with a uniform resource locator to said second control agent node and from resources available on an intranet coupled to said second control program agent node.

25. A computer network according to claim 23,

wherein said second control program agent node is coupled via a network with facilities provided within an intranet for private owner facilities and which may be protected by firewalls at the intranet boundary,

said second control program agent node located somewhere on the Internet supporting said second control program agent by a coupling or addressing with a uniform resource locator,

said first control program agent being coupled to said second control program agent node located somewhere on the Internet supporting said second control program agent and coupled to and supporting a a command file server, said command file server being coupled to a database gateway for gathering information from databases coupled to said database gateway and located on different database servers, said command file server supporting a plurality of command file objects which are programmed to perform web browser service support functions at the request of a user of said web browser to access information within the intranet and to gather information located elsewhere via the Internet as a sub-agent of said control program agent.

26. A computer network according to claim 23,

wherein said web browser is at a web server location with said web server providing said control program agent node, and browser requests, if authorized for access across said intranet, accesses a command file agent in a web server on said intranet providing said second command file agent node, which then

utilize DIS capsules provided by a DIS Server functioning as a command file server.